

CAPITULO I

ELEMENTOS DEL OBJETO DE CONOCIMIENTO

1.-El Problema.-

A lo largo de la historia los elementos tridimensionales y espaciales se han convertido en un tipo de estructura fundamental en la ingeniería y su empleo es muy amplio en campos de la carrera de ingeniería civil. Algunas de sus funciones más usuales son como estructura de edificios, hospitales, escuelas o en muros de contención. Dada la complejidad del funcionamiento de los elementos estructurales en tres dimensiones, se hace imprescindible el análisis del comportamiento de estos elementos.

El análisis del tipo de estructura seleccionado, debe hacerse con base a la idealización sobre cómo están apoyados y conectados los miembros entre sí, y la aplicación de cargas (fuerzas, cargas térmicas y reacciones) a lo largo del miembro, generando en la estructura un conjunto de fuerzas internas y deformaciones necesarias que el ingeniero estructural pueda diseñar considerando las recomendaciones establecidas en la normativa vigente, el dimensionamiento eficiente de los elementos, la calidad de los materiales, y el acero de refuerzo necesario.

Es importante analizar el comportamiento de una estructura cuando está sometida a carga, porque se requiere conocer los desplazamientos de los nudos, y las fuerzas internas y esfuerzo. Esas respuestas de la estructura son necesarias para el dimensionamiento adecuado de la sección transversal del elemento, la resistencia del concreto, resistencia del acero y la posición de los traslapes en los refuerzos longitudinales.

El método de elementos finitos es muy utilizado en diferentes campos de la ingeniería, mediante este método se obtiene representaciones de los fenómenos observados con mayor aproximación a la realidad por lo que suele emplearse para el desarrollo de programas que pueden simular el evento estudiado.

Asimismo el problema es averiguar en qué medida se ajusta el método de elementos finitos al estudio del análisis estructural de elementos articulados en tres

dimensiones (pórticos en el espacio) y por qué los software comerciales trabajan con dicho método.

1.1.-Planteamiento.-

Son muchas las áreas de la ingeniería en donde se necesita determinar los valores de tensiones y deformaciones de sólidos continuos sometidos a esfuerzos. Las soluciones analíticas exactas de las ecuaciones que permiten encontrar estos valores en cualquier punto de una pieza, son aplicables a casos sencillos, partiendo de hipótesis y simplificaciones basadas en los principios de la estática y resistencia de materiales.

Pero, cuando los problemas se tornan más complejos en su geometría y estados de cargas, la posibilidad de obtener ecuaciones sencillas que reflejen la solución se ve seriamente limitada.

La teoría de los elementos finitos proporcionó resultados a problemas más complejos basándose en la subdivisión de un dominio en subdominios o elementos de geometría simple, que permite el tratamiento analítico e individual de cada elemento, para que en una etapa posterior sean ensamblados y dar una solución al conjunto. Se propone un proyecto sobre uno de los más importantes métodos computacionales en Ingeniería: El Método de los Elementos Finitos (MEF), pero aplicado en el campo de la Ingeniería Civil y más concretamente al Análisis Estructural de estructuras.

1.2.-Objetivos.-

1.2.1.- Objetivo General.-

- ✚ Diseñar un software académico para la solución de pórticos en el espacio utilizando el método de elementos finitos a través del lenguaje de programación MATLAB de acuerdo al alcance establecido del proyecto.

1.2.2.- Objetivo Específicos.-

- ✚ Analizar en qué medida se ajustan los resultados arrojados por dicho método al utilizado por el software comercial SAP 2000 16.
- ✚ Implementar un algoritmo que permita encontrar la solución del sistema, de manera eficiente y que sea capaz de resolver matrices de gran tamaño, sin afectar el rendimiento normal de la computadora.
- ✚ Implementar el uso sistemas de unidades, para entrada y salida de dato.

- ✚ Determinar las deformaciones y reacciones del sistema estructural en el espacio, de acuerdo a las solicitaciones de cargas en miembros y nodos.
- ✚ Programar el software computarizado M.E.F. en idioma español utilizando el lenguaje de programación Matlab, más específicamente su entorno gráfico denominado Guide.
- ✚ Aportar a la Universidad Autónoma Juan Misael Saracho el software computarizado M.E.F. de forma gratuita junto con su código de programación abierto y un tutorial de manejo de dicho software.
- ✚ Incentivar el desarrollo y ampliación del programa mencionado así como también incentivar la creación de nuevos programas mediante elementos finitos en distintos campos que puedan ser utilizados por alumnos de la Universidad Autónoma Juan Misael Saracho aprovechando cursos que se vayan dando para poder despertar la curiosidad de los estudiantes para poder estudiar más a fondo este método universal de cálculo.

1.3.- Justificación.-

Las razones por las cuales se elabora el perfil de proyecto de grado son las siguientes:

- ✚ Profundizar conocimientos adquiridos en el diseño de un software por el Método de Elementos Finitos para solución de pórticos espaciales para esfuerzos internos y externos..
- ✚ El método de los elementos finitos, es un procedimiento basado en técnicas computacionales, que puede ser usado para analizar estructuras y diferentes sistemas continuos. Es un método numérico versátil, y que es ampliamente aplicado para resolver problemas que cubren casi todo el espectro de análisis ingenieriles. Sus aplicaciones comunes, incluyen el comportamiento de sistemas estáticos, dinámicos y térmicos. Los avances en el hardware, han facilitado y aumentado la eficiencia del software de elementos finitos, para la solución de sistemas complejos de ingeniería sobre computadores personales.
- ✚ Los avances tecnológicos obligan al conocimiento cabal de los métodos más precisos para el cálculo en la ingeniería, donde el error cada día se vuelve

mínimo, por lo tanto es necesario el conocimiento analítico y computacional de teorías de cálculo más precisas y exactas, es deber del ingeniero civil el conocer en su totalidad estos métodos modernos para poder realizar aportes que puedan ayudar a simplificar problemas que se dan en la vida laboral.

✚ Contribuir una adecuada solución para un mejor entendimiento del estudiante sobre

los métodos matriciales, así también en la aplicación práctica que tienen dichos métodos al realizar programas informáticos con algoritmos sencillos para la realización de software que faciliten el cálculo de estructuras en ingeniería civil.

1.4.- Marco Conceptual.-

1.4.1-Espacial.-

El presente trabajo de aplicación tendrá lugar en la Ciudad de Tarija-Provincia Cercado del Departamento de Tarija.

1.4.2.-Temporal.-

El siguiente trabajo de investigación se realizara durante el segundo periodo de gestión académica 2016 y puede servir de base a futuros proyectos interesados en el objeto de lenguajes de programación a través de dicho método.

1.5.- Alcance de la propuesta-

1.5.1.- Tipo de estudio

Estudio investigativo, donde se evaluara la diferente bibliografía para realizar un análisis y diseño de un software práctico que permita la resolución de pórticos en el espacio utilizando el método de los elementos finitos.

1.5.2.-Alcance del proyecto

El estudio investigativo tendrá los siguientes alcances:

✚ Solución de pórticos en el espacio para el resultado de esfuerzos internos (momentos, cortantes, normales, desplazamientos, ángulos de giro) y esfuerzos externos (reacciones).

- ✚ La herramienta de programación Matlab que tiene su propio código y lenguaje de programación.
- ✚ Solución de los casos planteados (carga uniforme, triangular y puntual), de manera estática

1.5.3.-Premisas

Como parte del método de este trabajo, se plantearan las siguientes premisas que ayudaran

en la simplificación del problema:

- ✚ El material de la pieza es homogéneo.
- ✚ El material es isótropo.
- ✚ Las cargas son estáticas.

1.5.4.-Hipótesis

El método de elementos finitos puede programarse computacionalmente para diseñar un software académico a través del lenguaje de programación MAT LAB, que determine resultados confiables, comprobados, realizando una comparación con el software comercial SAP 2000 16 dando resultados que alcancen $\pm 10\%$ de precisión.

CAPITULO II

MARCO TEORICO

2.1- Método de los elementos finitos.-

2.1.1.-Breve historia de los elementos finitos.-

Los Conceptos de discretización numérica para resolver problemas de Ciencia e Ingeniería son la base para la formulación del método de elemento finito. La aproximación geométrica más antigua lleva a las pirámides egipcias de 5.000 años. Por otro lado la aproximación numérica podría registrarse históricamente en China, Egipto y Grecia.

Los registros muestran que los chinos calcularon el valor aproximado de π en el primer siglo de nuestra era, con un valor de 3.1547 siendo usado para calcular el volumen de un cilindro.

En el segundo siglo E.C. el astrónomo Chang Heng aproximó el valor de π como 3.146614245 en la Dinastía oriental de Jihn (265-317 E.C) en su comentario de matemáticas usó un polígono regular inscrito en una circunferencia para poder aproximar π la cual halló un valor de 3.1416 (3927/1250); es interesante notar que el uso un polígono de 3072 lados, es decir elementos finitos. De acuerdo con el manuscrito Ahmes, se muestra que para 1.500 A.C., los Egipcios usaban como valor de $\pi = 3,1416$. Un papiro de tiempos más tempranos, ahora en Moscú, indica que los egipcios usaron la fórmula para el volumen de una pirámide y el área de un círculo de manera aproximada; en 1.800 A. C. Arquímedes usó el concepto de elementos finitos para calcular volúmenes.

En el contexto estructural, las soluciones tanto en elasticidad como en análisis estructural tuvieron un inicio del Método del Elemento Finito con Timoshenko, pero si se considera que el análisis de marcos establece el inicio del método del Elemento Finito, entonces los pioneros fueron Castigliano, Mhor y Maxwell, entre otros, en el periodo 1850-1875.

En 1915, Maney de los Estados Unidos de Norteamérica, presentó el método pendiente-deformación, expresando los momentos en términos de desplazamientos lineales y angulares en los nodos de la estructura, lo cual es una de las formulaciones

para plantear el método de las rigideces y un desarrollo similar, fue planteado por Ostenfeld en Dinamarca.

En el año 1929, Hardy Cross hizo público un método para analizar marcos basado en distribuciones angulares, el cual se utilizó por los siguientes 35 años. En forma paralela a los primeros trabajos sobre análisis de estructuras reticulares, se resolvieron problemas de mecánica del medio continuo usando una analogía con estructuras formadas por barras diagonales para generar mallas con elementos triangulares. A principios de los años cuarenta Courant propuso funciones de interpolación polinomiales por secciones para formular sub regiones triangulares como un caso especial del método variacional de Rayleigh-Ritz, que obtiene soluciones aproximadas. Actualmente, el método del elemento finito es utilizado con la ayuda de las computadoras, lo cual ha contribuido a su desarrollo al mismo ritmo que las computadoras.

Las publicaciones clásicas por Argyris y Kelsey a mediados de los 50, hicieron surgir los conceptos de análisis de marcos, discretizando no sólo en nodos, sino además en puntos intermedios de las barras y análisis de un continuo, lo que marcó un crecimiento explosivo en el método del elemento finito.

Basándose en el planteamiento estático del elemento finito, se han ampliado las aplicaciones que incluyen diversos efectos físicos y vibraciones en el Análisis dinámico, pandeo y post-pandeo, no linealidades en la geometría y en el material, efectos térmicos, interacción entre fluidos y estructuras, Aero elasticidad, interacción acústica-estructura, teoría de la fractura, estructuras laminadas, propagación de oleaje, dinámica estructural, respuesta dinámica aleatoria y muchas más aplicaciones.

Como una consecuencia de tantos campos de estudio, el uso de los programas de computadora orientados a cada caso, se han convertido en una práctica en los sitios involucrados en el análisis estructural.

En este método el cuerpo se divide en cierto número de “elementos” o “dimensiones finitas”, de ahí su nombre. Si el cuerpo tiene n ($n= 1,2,3$) dimensiones en el espacio, se dividirá en elementos finitos de n dimensiones.

Los cuerpos unidimensionales se dividen en elementos finitos mediante nodos.

Nodos

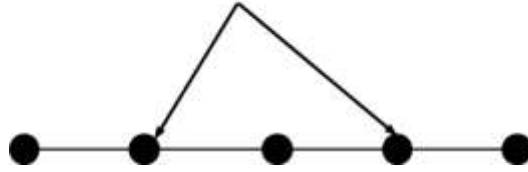


Ilustración 1 División de un cuerpo unidimensional mediante nodos.

Para la división de los cuerpos de dos y tres dimensiones se usan líneas y planos.

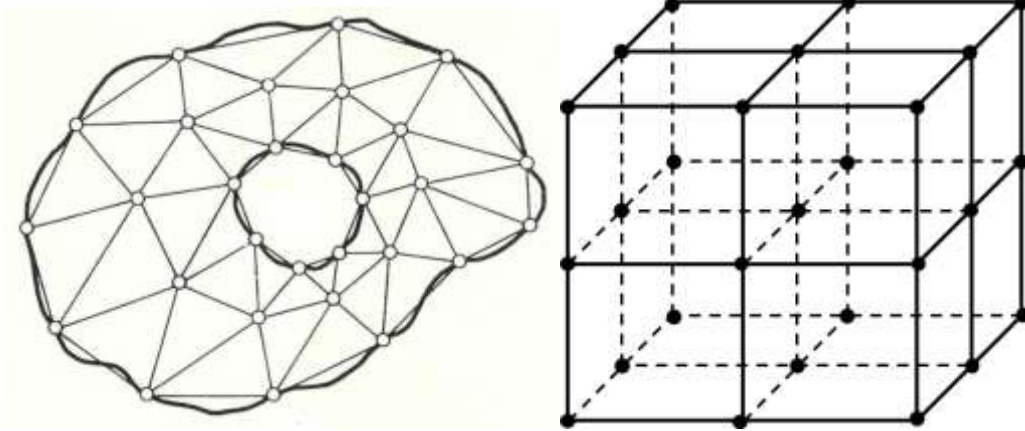


Ilustración 2 Ejemplo de malla bidimensional y tridimensional.

En los cuerpos unidimensionales los elementos finitos resultantes pueden tener longitudes diferentes, así mismo los elementos finitos en los cuerpos de dos y tres dimensiones pueden tener diferentes tamaño y forma. En todos los casos los elementos finitos estarán conectados por nodos. De esta manera el cuerpo objeto del estudio es sustituido por un sistema de elementos finitos conectados por nodos.

Una vez se tiene el cuerpo dividido en elementos finitos el siguiente paso es determinar la matriz de rigidez de los elementos individualmente. Luego estas se unen para formar la matriz de rigidez del cuerpo de manera que la continuidad de los movimientos y el equilibrio de fuerzas prevalezca en todos los nodos del modelo. Esto nos lleva a la siguiente ecuación matricial:

$$[K](\delta) = (P)$$

Donde $[K]$ es la matriz de rigidez del cuerpo, (P) es el vector de las fuerzas externas aplicadas en todos los nodos y (d) es el vector de los desplazamientos de los nodos. Para unas condiciones de contorno específicas y unas fuerzas aplicadas sobre el cuerpo esta ecuación se puede resolver hallando los desplazamientos de los nodos. Una vez hallados se pueden calcular las tensiones y los esfuerzos.

Como se ha dicho anteriormente $[K]$ se forma a partir de las matrices de rigidez de cada uno de los elementos $[K]$ en que se divide el cuerpo objeto del estudio. Para averiguar la $[K_i]$ de un elemento se estudian los desplazamientos en los vértices de ese elemento. En este caso se empleará el elemento rectangular, puesto que va a ser el que se utilice posteriormente para resolver los casos propuestos como ejemplos.

Igualmente en los cuerpos tridimensionales el tetraedro es mejor que el prisma rectangular. Otras veces dependiendo de la forma del cuerpo lo mejor es hacer una división mixta, es decir, con elementos de diferentes formas.

2.1.2.-Aplicaciones del método.-

Existen gran número de estructuras que su paso de revisión, tanto como post y pre proceso de diseño, son sometidos a rigurosos procesos de evaluación, tales como verificación de cortantes y flectores y saber si el diseño dado es correcto.

2.1.2.1.-Aplicación en mecánica de fluidos.-

El método de elementos finitos también puede ser usado en el análisis de mecánica de fluidos, librerías en python como ECOASTER o el Fenics o programas como Abaqus, Nastran nos pueden dar una idea de cómo este elemento físico se mueve e interacciona con el mundo aplicando soluciones particulares a la formulación de la ecuación general de los fluidos Navier y Stokes, idealizaciones de presas, así como simulaciones de comportamiento de las turbulencias de los fluidos en turbinas hidráulicas.

2.1.2.2.-Aplicaciones al análisis estructural.-

En programas como el Sap2000, el staad pro, tanto en el análisis de estructuras como en el análisis de suelo-estructura, así como el Safe o el programa Plaxis que son de uso para el cálculo por elementos finitos.

2.1.3.-Análisis de vigas.-

El análisis de vigas para el estudio de elementos finitos, se hace para entender en forma elemental los diferentes tipos de discretizaciones a elementos de simples a complejos, ya que para poder entender los análisis y cálculo de placas y sólidos de revolución es necesario entender los diferentes ítems del estudio de vigas, tanto sometidas a fuerzas axiales como a flexiones, que pueden ser tratadas por teorías muy conocidas como Bernoulli y Timoshenko, sobre todo entendimiento de condiciones de contorno para pasar a estadios más complejos.

2.1.3.1.-Análisis de flexión de vigas.-

El clásico problema de vigas puede ser resuelto con el método tradicional de Resistencia de materiales, sin embargo resolver el problema con el método sofisticado del MEF es de gran interés didáctico, pues en este particular problema cada nodo puede ser trabajado con dos variables y pueden servir como conceptos primarios para estudio de placas y láminas.

Entre estos principios y conceptos básicos existen dos formas planteadas como estudio generalizado de las mismas, como son el estudio de vigas por el método Viga Bernoulli y el estudio de vigas por el método Viga Timoshenko.

2.1.3.2.-Deformación en vigas.-

El análisis estructural de las vigas suele dividirse en vigas isostáticas e hiperestáticas. Recordemos que esta división corresponde a las condiciones de apoyo que presente el elemento a analizar. Si la viga tiene un número igual o inferior a tres incógnitas en sus reacciones, bastará con aplicar las condiciones de equilibrio estático para resolverla.

$$\sum F_x = 0 \quad \sum F_y = 0 \quad \sum M = 0$$

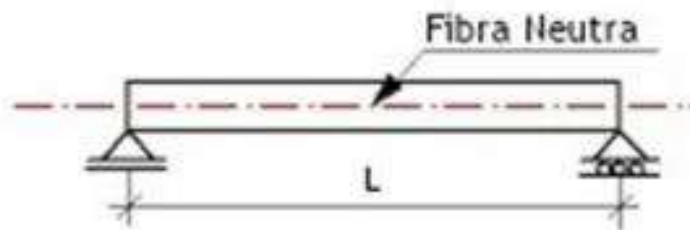
Si en cambio, la viga presenta un mayor número de incógnitas, no bastará con las ecuaciones antes indicadas, sino que será necesario incorporar nuevas expresiones. Para abordar el análisis de las vigas hiperestáticas o estáticamente indeterminadas resulta necesario analizar las deformaciones que experimentará la viga, luego de ser

cargada. Las distintas cargas sobre la viga generan tensiones de corte y flexión en la barra y, a su vez, la hacen deformarse.

El análisis de las deformaciones tiene básicamente dos objetivos. Por una parte, el poder obtener nuevas condiciones, que traducidas en ecuaciones, nos permitan resolver las incógnitas en vigas hiperestáticas. Y por otra parte, las deformaciones en sí, deben ser limitadas. Los envigados de madera o acero, por ejemplo, pueden quedar correctamente diseñados por resistencia, vale decir, no se romperán bajo la carga, pero podrán deformarse más allá de lo deseable, lo que llevaría consigo el colapso de elementos de terminación como cielos falsos o ventanales. No resulta extraño entonces que muchos dimensionamientos queden determinados por la deformación y no por la resistencia.

2.1.3.3.-Línea elástica.-

Denominaremos línea elástica a la curva que forma la fibra neutra una vez cargada la viga, considerando que ésta se encontraba inicialmente recta.



Ilustracion 3. Fibra neutra

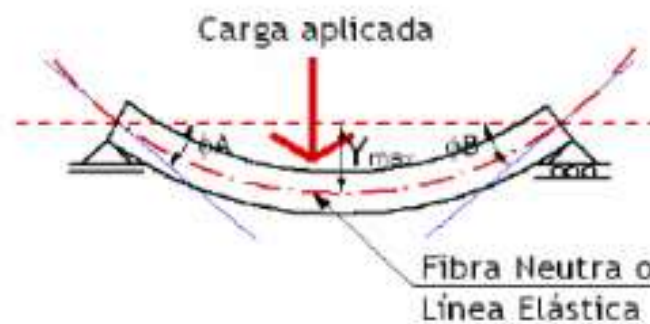


Ilustración 4.Fibra neutra con una carga aplicada

2.1.3.4.- Supuesto base.-

Para establecer una serie de relaciones al interior de la sección, indicamos que se trata de una viga, cuyo material se encuentra solicitado dentro del rango de proporcionalidad entre tensiones y deformaciones, en donde se admite la conservación de las caras planas. Dicho en otra forma, donde se cumplen la ley de Hooke y la hipótesis de Bernoulli-Navier.

Ley de Hooke

Establece que la relación entre la tensión y la deformación unitaria es una constante y se denomina módulo de elasticidad.

$$E = \frac{\tau}{e} \quad \rightarrow \quad \tau = E * e$$

Siendo:




E: Elasticidad

e: La deformación unitaria

τ : Tensión

2.1.3.5.- Método de análisis en la deformación de vigas.-

Existen diferentes métodos para abordar el análisis de las deformaciones en la viga:

-  Método de Área de Momentos.
-  Método de Doble Integración.
-  Método de la Viga Conjugada.

Si bien, todos presentan su mecánica propia, a la vez tienen una partida común, que es justamente el análisis de la elástica, expuesto anteriormente.

En este caso, para la resolución de los problemas planteados, profundizaremos en el método de la doble integración.

en la industria y en el sector social. El método de equilibrio al Análisis de Estructuras se desarrolló a principios del siglo XX y desplazaron progresivamente a los primeros en las aplicaciones prácticas. A partir de la segunda mitad del siglo XX, la utilización del ordenador digital produce una rápida evolución en la investigación de muchas ramas de la ciencia y de la técnica, dando lugar a procedimientos "numéricos", adecuados para el uso de los mismos.

En el campo del Análisis de Estructuras el ordenador ha llevado de forma natural al desarrollo del cálculo matricial de estructuras. Paralelamente, se desarrollan los métodos de aproximación numérica discreta tales como los métodos de las diferencias finitas y de los elementos finitos, que permiten resolver problemas mecánicos en estructuras continuas y cuya aplicación se extiende, incluso, a la resolución de problemas no lineales.

A finales del siglo XX, la rápida generalización del uso de los ordenadores personales hace de estos la herramienta básica de cálculo en ingeniería; los métodos de cálculo de estructuras por ordenador son hoy, un elemento esencial en la enseñanza de la Mecánica de Estructuras. La aplicación de estos métodos permite:

- ✚ Formular una metodología de análisis compacta y basada en principios generales,
- ✚ Desarrollar procedimientos prácticos de análisis y,
- ✚ Organizar de forma simple los programas de ordenador de Cálculo de Estructuras.

Por otro lado, debe decirse que los métodos matriciales se caracterizan por una gran cantidad de cálculos sistemáticos, por lo que su aplicación se basa en la utilización del ordenador y no en el cálculo manual. Son, por lo tanto, métodos de análisis adecuados para estructuras complejas. En el caso de problemas sencillos, de fácil resolución por métodos manuales, los métodos matriciales no aportan ninguna ventaja importante.

Tanto los Métodos de Compatibilidad como los de Equilibrio pueden plantearse de cara a su resolución automática, dando lugar a los Métodos de Flexibilidad y de Rigidez, respectivamente. Sin embargo, los segundos cuentan con la importante

ventaja sobre los primeros de que su formulación es más sistemática. En la práctica, la casi totalidad de los programas modernos de Cálculo de Estructuras, ya sean éstas de piezas o continuas, se basan en el Método de Rigidez.

2.2.1.-Bases del método.- El método de la rigidez se basa en los tres principios fundamentales de la mecánica de estructuras: compatibilidad, equilibrio linealidad/superposición.

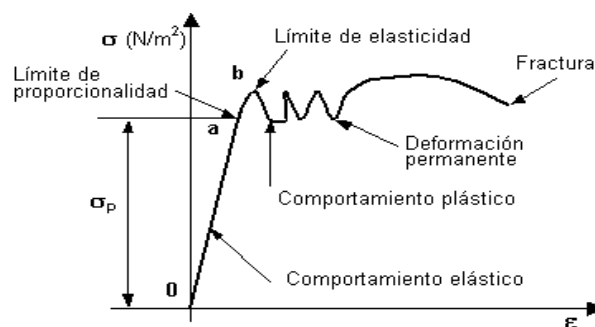
2.2.1.1.-Compatibilidad.-La deformación es una función continua y tiene un valor único en cada punto. En consecuencia, los movimientos también lo son y, en particular, los movimientos en los extremos de las piezas que concurren en un mismo nudo son idénticos para todas las piezas.

2.2.1.2.-Equilibrio.-Tanto la estructura globalmente como cada parte de la misma y en particular, cada nudo y pieza de la misma están en equilibrio estático, bajo la acción de las fuerzas exteriores y de los esfuerzos internos.

2.2.1.3.-Linealidad y superposición.

2.2.1.3.1.-Linealidad.-Esta hipótesis supone que la ley del comportamiento del material es lineal. Esto quiere decir que si la carga aplicada sobre el material se multiplica por cierto valor, las tensiones y deformaciones resultantes vendrán multiplicadas por ese mismo valor. El comportamiento de un material puede clasificarse de varias maneras. En primer lugar puede ser, elástico o inelástico; en segundo lugar puede ser lineal o no lineal. Para que se cumpla la hipótesis de linealidad del comportamiento del material, el material debe estar trabajando en la zona lineal.

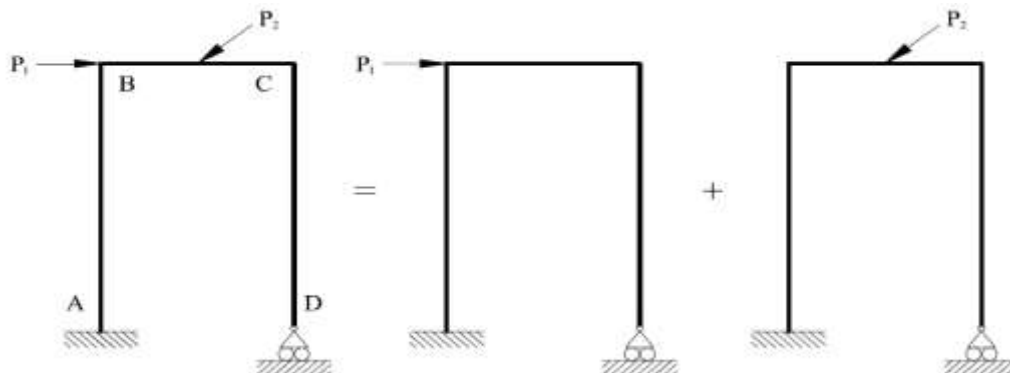
Ilustración 5. Diagrama tensión-deformación de un material.



2.2.1.3.2.-Principio de superposición.- El principio de superposición constituye la base de gran parte de la teoría del análisis estructural. Puede enunciarse como sigue: El desplazamiento o esfuerzo total en un punto de una estructura sometida a varias cargas se puede determinar sumando los desplazamientos o esfuerzos que ocasiona cada una de las cargas que actúan por separado. Para que esto sea válido, es necesario que exista una relación lineal entre las cargas, esfuerzos y desplazamientos.

- ✚ El material debe comportarse de manera elástica lineal, a fin de que sea válida la ley de Hooke y la carga sea proporcional al desplazamiento
- ✚ La geometría de la estructura no debe sufrir cambios importantes cuando se aplican las cargas. Si los desplazamientos son grandes, entonces cambian considerablemente la posición y orientación de las cargas. Un ejemplo es el caso de una columna sometida a una carga de pandeo

Ilustración 6. Principio de superposición



2.3.-Analizar una estructura según el método de la rigidez.-

Los pasos para resolver una estructura según el Método de Rigidez. El esquema de resolución se muestra, detallado, en el esquema de la **Ilustración 7**. Como se observa, el proceso secuencial consiste en:

1. Definir la geometría de la estructura y las acciones, así como las condiciones de apoyo.
2. Identificar el número de movimientos incógnita que determinan la deformación de la estructura, a base de considerar las correspondientes condiciones de compatibilidad en los nudos.
3. Resolver las piezas individuales, en función de los movimientos de sus extremos, a base de satisfacer las condiciones de equilibrio y compatibilidad en las piezas.
4. Imponer las condiciones de equilibrio en los nudos.
5. Imponer las condiciones de apoyo de la estructura.
6. Determinar los movimientos incógnita, a base de resolver el sistema de ecuaciones resultante.
7. Determinar los esfuerzos y reacciones en la estructura.

2.3.1.-Grados de libertad.-

Grados de libertad: Son los desplazamientos independientes (traslaciones y rotaciones) de los nodos que son necesarios para especificar la forma deformada de una estructura, cuando se vayan a sujetar a una carga arbitraria.

2.3.2.-Estudio matricial.-

En el modelo analítico de una estructura:

- ✚ Los nodos se cuentan con un número dentro de un círculo (inicia con nodo libre). El orden en que se enumeren los nodos indica el sentido que se da al elemento.
- ✚ Los elementos se cuentan con su número escrito dentro de un rectángulo. El sentido del mismo se define desde el nodo con el menor número (nodo inicial) hacia aquel que tenga el mayor número (nodo final).

- ✚ Los grados de libertad se representan por flechas rectas (si es para traslación) o flechas curvas (si es para rotación) siempre en sentido positivo. A cada grado de libertad restringido por alguna reacción, corresponde una fuerza o momento, según sea el caso.
- ✚ Al numerar los grados de libertad, el primer número es para la dirección X, el segundo para la dirección Y y el tercero en dirección Z.

2.3.3.-Coordenadas locales.-

Son llamadas también coordenadas particulares o del elemento. Se denominan así debido a que respecto a éstas se referencian todas las propiedades de los elementos, como las dimensiones y momentos de inercia, al igual que las cargas aplicadas sobre los mismos y las fuerzas internas a que se ven sometidos. Se definen colocando el eje x a lo largo del eje centroidal del elemento, colocando el origen del mismo en el nodo inicial. Los demás ejes (y, z) se definen teniendo en cuenta la ortogonalidad de los mismos. Con estas coordenadas queda definida la orientación del elemento estructural.

2.3.4.-Transformación de coordenadas.-

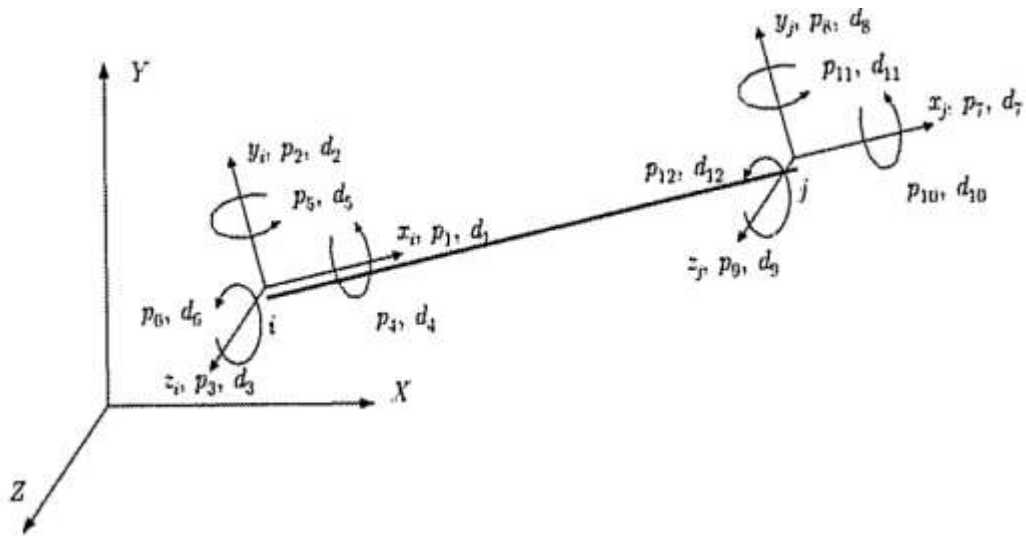
Cuando los miembros de una estructura están orientados en direcciones diferentes es

necesario transformar las relaciones de rigidez de cada miembro, del sistema de coordenadas locales del mismo, hacia una sistema común de coordenadas globales. Luego se combinan las relaciones de rigidez de los miembros así obtenidas a fin de establecer las relaciones de rigidez para la estructura completa.

Dependiendo del tipo de elemento estructural, se obtendrá una matriz de transformación diferente.

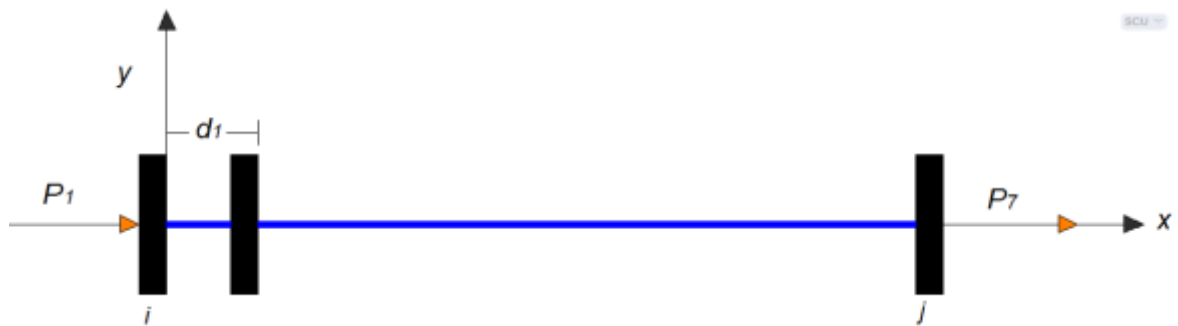
2.3.5.-Rigideces elementales.-

Ilustración 8. Sistemas de coordenadas global y local para elementos espaciales de nudos rígidos.



2.3.5.1.-- Rigideces para desplazamientos según el eje x.-

Ilustración 9. Rigidez para desplazamiento según el eje x, en el extremo i



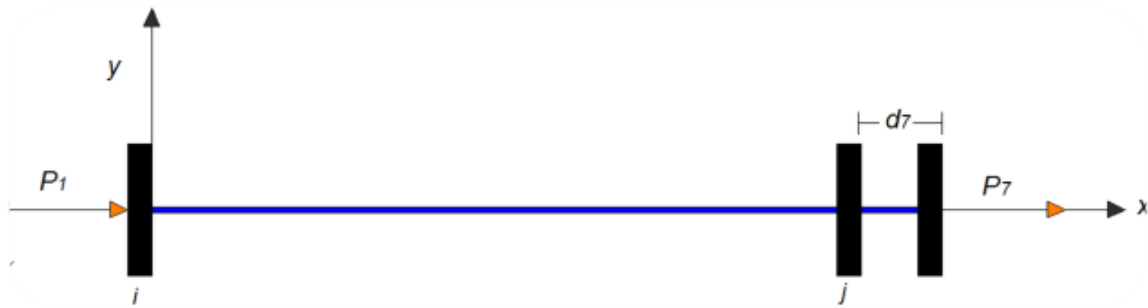
Si le aplicamos un desplazamiento d_1 en el extremo i el esfuerzo es:

$$P_1 = \frac{EA}{L} d_1 \quad P_7 = -\frac{EA}{L} d_1$$

Y las rigideces correspondientes:

$$K_{1,1} = \frac{EA}{L} \quad K_{7,1} = -\frac{EA}{L}$$

Ilustración 10. Rigidez para desplazamiento según el eje x, en el extremo j



Si le aplicamos un desplazamiento d_7 en el extremo j el esfuerzo es:

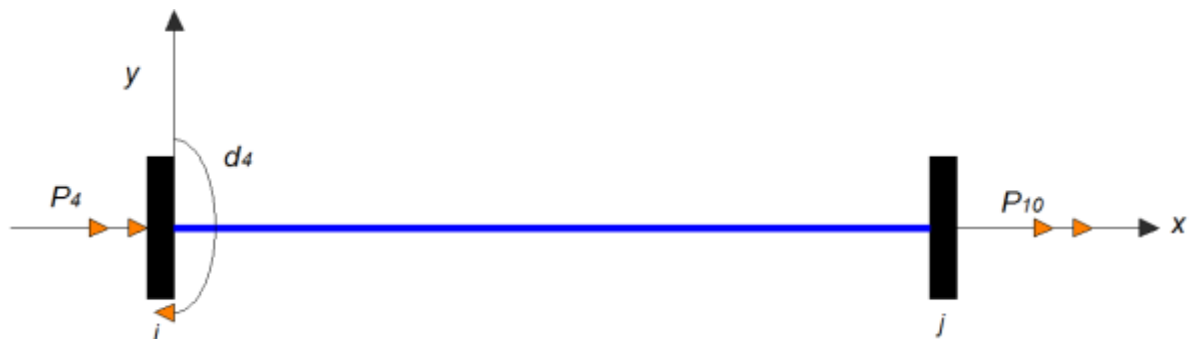
$$P_1 = -\frac{EA}{L} d_7 \quad P_7 = \frac{EA}{L} d_7$$

Y las rigideces correspondientes:

$$K_{1,7} = -\frac{EA}{L} \quad K_{7,7} = \frac{EA}{L}$$

2.3.5.2.- Rigideces para giros según el eje x.-

Ilustración 11. Rigidez para giro según el eje x, en el extremo



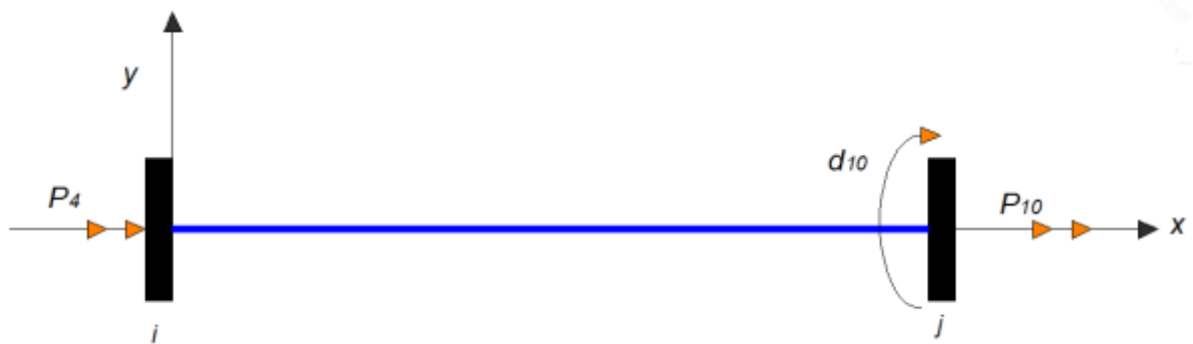
El esfuerzo debido al giro d_4 en el extremo i

$$P_4 = \frac{GJ}{L} d_4 \quad P_{10} = -\frac{GJ}{L} d_4$$

Y la rigideces correspondientes

$$K_{4.4} = \frac{GJ}{L} \quad K_{10.4} = -\frac{GJ}{L}$$

Ilustración 12. Rigidez para giro según el eje x, en el extremo j.



El esfuerzo debido al giro d_{10} en el extremo i

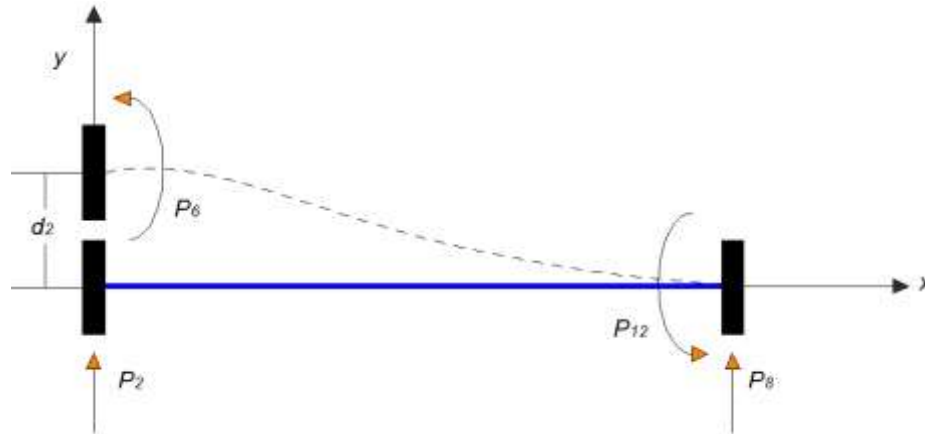
$$P_4 = -\frac{GJ}{L} d_{10} \quad P_{10} = \frac{GJ}{L} d_{10}$$

Y la rigideces correspondientes

$$K_{4.10} = -\frac{GJ}{L} \quad K_{10.10} = \frac{GJ}{L}$$

2.3.5.3.-Rigideces para desplazamientos según el eje y.-

Ilustración 13. Rigidez para desplazamiento según el eje y, en el extremo i

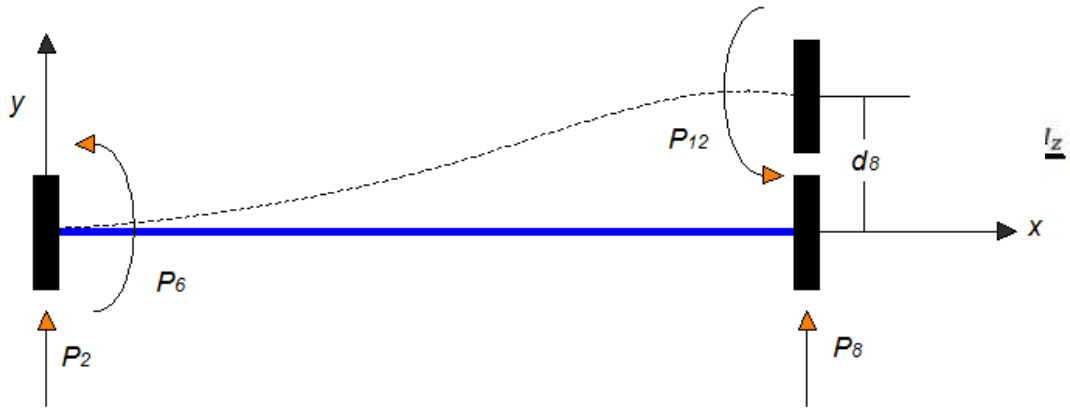


Los esfuerzos correspondientes debido al desplazamiento d_2 en el extremo i son:

$$P_6 = \frac{6EI_z}{L^2} d_2 \quad P_{12} = \frac{6EI_z}{L^2} d_2 \quad P_2 = \frac{12EI_z}{L^3} d_2 \quad P_8 = -\frac{12EI_z}{L^3} d_2$$

Y las rigideces correspondientes:

Ilustración 14. Rigidez para desplazamiento según el eje y , en el extremo j



Los esfuerzos correspondientes debido al desplazamiento d_8 en el extremo j son:

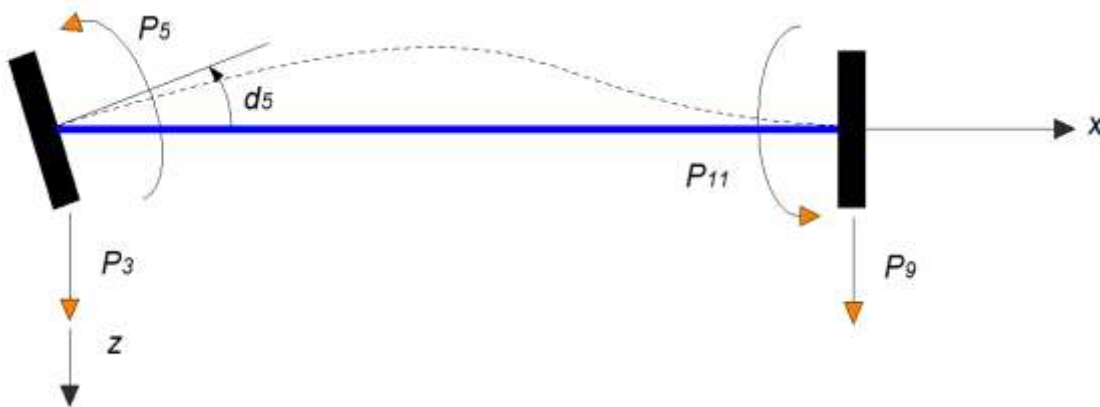
$$P_6 = -\frac{6EI_z}{L^2} d_8 \quad P_{12} = -\frac{6EI_z}{L^2} d_8 \quad P_2 = -\frac{12EI_z}{L^3} d_8 \quad P_8 = \frac{12EI_z}{L^3} d_8$$

Y las rigideces correspondientes:

$$K_{6.8} = -\frac{6EI_z}{L^2} \quad K_{12.8} = -\frac{6EI_z}{L^2} \quad K_{2.8} = -\frac{12EI_z}{L^3} \quad K_{8.8} = \frac{12EI_z}{L^3}$$

2.3.5.4.- Rigideces para giros según el eje y

Ilustración 15. Rigidez para giro según el eje y, en el extremo i.



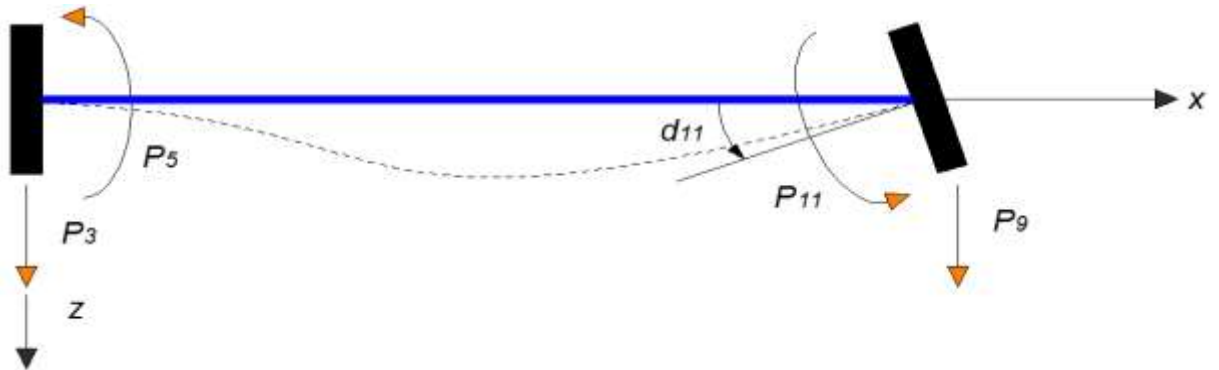
El esfuerzo debido al giro d_5 en el extremo i

$$P_5 = \frac{4EI_y}{L} d_5 \quad P_{11} = \frac{2EI_y}{L} d_5 \quad P_3 = -\frac{6EI_y}{L^2} d_5 \quad P_9 = \frac{6EI_y}{L} d_5$$

Y la rigideces correspondientes

$$K_{5.5} = \frac{4EI_y}{L} \quad K_{11.5} = \frac{2EI_y}{L} \quad K_{3.5} = -\frac{6EI_y}{L^2} \quad K_{9.5} = -\frac{6EI_y}{L^2}$$

Ilustración 16. Rigidez para giro según el eje y, en el extremo j.



El esfuerzo debido al giro d_{11} en el extremo j

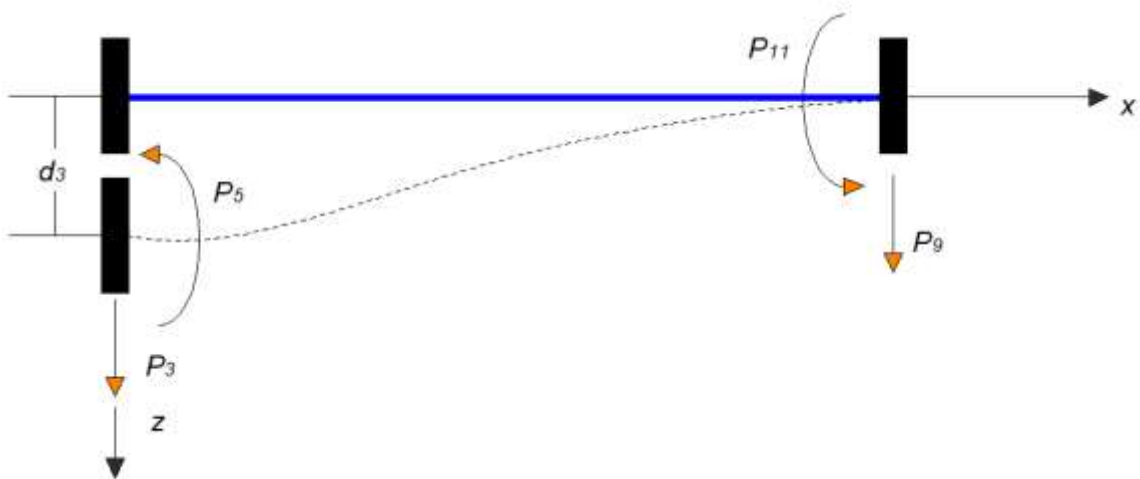
$$P_5 = \frac{2EI_y}{L} d_{11} \quad P_{11} = \frac{4EI_y}{L} d_{11} \quad P_3 = -\frac{6EI_y}{L^2} d_{11} \quad P_9 = \frac{6EI_y}{L^2} d_{11}$$

Y la rigideces correspondientes:

$$K_{5.11} = \frac{2EI_y}{L} \quad K_{11.11} = \frac{4EI_y}{L} \quad K_{3.11} = -\frac{6EI_y}{L^2} \quad K_{9.11} = \frac{6EI_y}{L^2}$$

2.3.5.5.- Rigideces para desplazamientos según el eje z.-

Ilustración 17. Rigidez para desplazamiento según el eje z, en el extremo i.



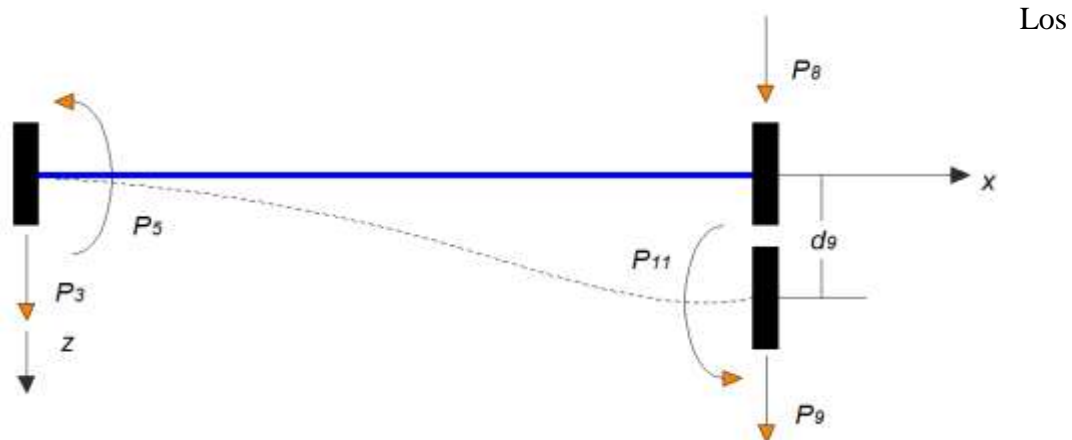
Los esfuerzos correspondientes debido al desplazamiento d_3 en el extremo i son:

$$P_3 = \frac{12EI_y}{L^3} d_3 \quad P_9 = -\frac{12EI_y}{L^3} d_3 \quad P_5 = -\frac{6EI_y}{L^2} d_3 \quad P_{11} = -\frac{6EI_y}{L^2} d_3$$

Y las rigideces correspondientes:

$$K_{3,3} = \frac{12EI_y}{L^3} \quad K_{9,3} = -\frac{12EI_y}{L^3} \quad K_{5,3} = -\frac{6EI_y}{L^2} \quad K_{11,3} = -\frac{6EI_y}{L^2}$$

Ilustración 18. Rigidez para desplazamiento según el eje z, en el extremo



esfuerzos correspondientes debido al desplazamiento d_9 en el extremo j son:

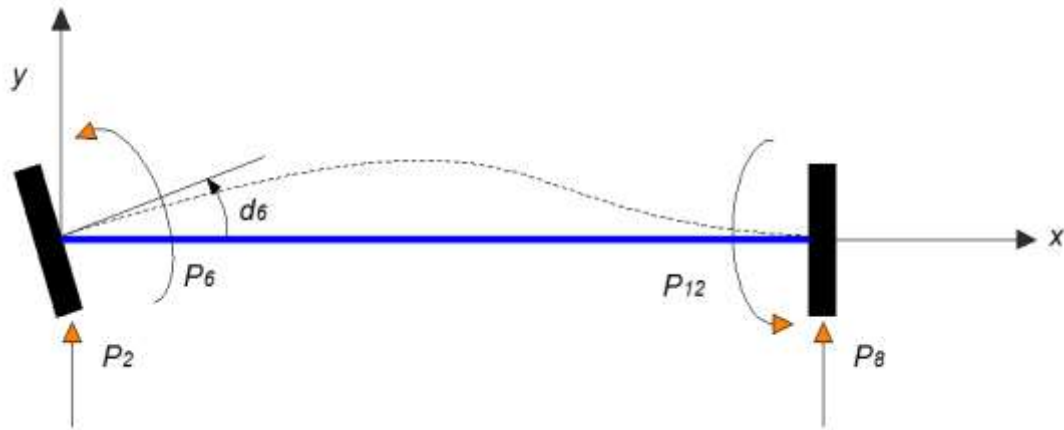
$$P_3 = -\frac{12EI_y}{L^3} d_9 \quad P_9 = \frac{2EI_y}{L^3} d_9 \quad P_5 = \frac{6EI_y}{L^2} d_9 \quad P_{11} = -\frac{6EI_y}{L^2} d_9$$

Y las rigideces correspondientes:

$$K_{3,9} = -\frac{12EI_y}{L^3} \quad K_{9,9} = \frac{2EI_y}{L^3} \quad K_{5,9} = \frac{6EI_y}{L^2} \quad K_{11,9} = -\frac{6EI_y}{L^2}$$

2.3.5.6.--Rigideces para giros según el eje z.-

Ilustración 19. Rigidez para giro según el eje z, en el extremo i.



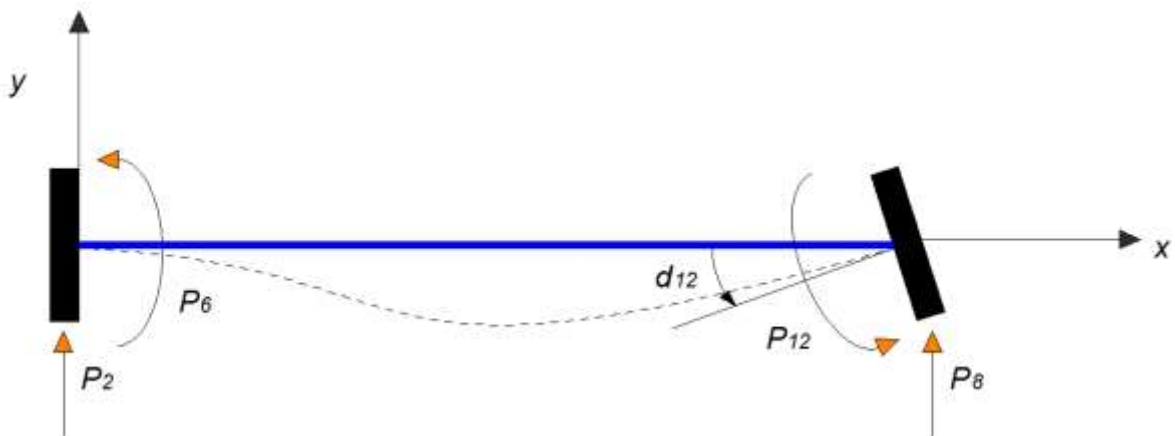
Los esfuerzos correspondientes debido al desplazamiento d_6 en el extremo i son:

$$P_6 = \frac{4EI_z}{L} d_6 \quad P_{12} = \frac{2EI_z}{L} d_6 \quad P_2 = \frac{6EI_z}{L^2} d_6 \quad P_8 = -\frac{6EI_z}{L^2} d_6$$

Y la rigideces correspondientes

$$K_{6.6} = \frac{4EI_z}{L} \quad K_{12.6} = \frac{2EI_z}{L} \quad K_{2.6} = \frac{6EI_z}{L^2} \quad K_{8.6} = -\frac{6EI_z}{L^2}$$

Ilustración 20. Rigidez para giro según el eje z, en el extremo j.



Los esfuerzos correspondientes debido al desplazamiento d_{12} en el extremo j son:

$$P_6 = \frac{4EI_z}{L} d_6 \quad P_{12} = \frac{2EI_z}{L} d_6 \quad P_2 = \frac{6EI_z}{L^2} d_6 \quad P_8 = -\frac{6EI_z}{L^2} d_6$$

Y la Rigideces correspondientes

$$K_{6,6} = \frac{4EI_z}{L} \quad K_{12,6} = \frac{2EI_z}{L} \quad K_{2,6} = \frac{6EI_z}{L^2} \quad K_{8,6} = -\frac{6EI_z}{L^2}$$

2.3.6.-Matriz de rigidez elemental.-

La matriz de rigidez de un elemento cuyo sistema de coordenadas coincide con los ejes principales de inercia de la sección.

$$\bar{\mathbf{K}} = \begin{bmatrix} K_{11} & K_{12} \\ K_{21} & K_{22} \end{bmatrix}$$

Donde:

$$K_{11} = \begin{bmatrix} \frac{AE}{L} & 0 & 0 & 0 & 0 & 0 \\ 0 & \frac{12EI_z}{L^3} & 0 & 0 & 0 & \frac{6EI_z}{L^2} \\ 0 & 0 & \frac{12EI_y}{L^3} & 0 & -\frac{6EI_y}{L^2} & 0 \\ 0 & 0 & 0 & \frac{GJ}{L} & 0 & 0 \\ 0 & 0 & -\frac{6EI_y}{L^2} & 0 & \frac{4EI_y}{L} & 0 \\ 0 & \frac{6EI_z}{L^2} & 0 & 0 & 0 & \frac{4EI_z}{L} \end{bmatrix}$$

$$K_{12} = \begin{bmatrix} -\frac{AE}{L} & 0 & 0 & 0 & 0 & 0 \\ 0 & -\frac{12EI_z}{L^3} & 0 & 0 & 0 & \frac{6EI_z}{L^2} \\ 0 & 0 & -\frac{12EI_y}{L^3} & 0 & -\frac{6EI_y}{L^2} & 0 \\ 0 & 0 & 0 & -\frac{GJ}{L} & 0 & 0 \\ 0 & 0 & \frac{6EI_y}{L^2} & 0 & \frac{2EI_y}{L} & 0 \\ 0 & -\frac{6EI_z}{L^2} & 0 & 0 & 0 & \frac{2EI_z}{L} \end{bmatrix}$$

$$K_{21} = \begin{bmatrix} -\frac{AE}{L} & 0 & 0 & 0 & 0 & 0 \\ 0 & -\frac{12EI_z}{L^3} & 0 & 0 & 0 & -\frac{6EI_z}{L^2} \\ 0 & 0 & -\frac{12EI_y}{L^3} & 0 & \frac{6EI_y}{L^2} & 0 \\ 0 & 0 & 0 & -\frac{GJ}{L} & 0 & 0 \\ 0 & 0 & -\frac{6EI_y}{L^2} & 0 & \frac{2EI_y}{L} & 0 \\ 0 & \frac{6EI_z}{L^2} & 0 & 0 & 0 & \frac{2EI_z}{L} \end{bmatrix}$$

$$K_{22} = \begin{bmatrix} \frac{AE}{L} & 0 & 0 & 0 & 0 & 0 \\ 0 & \frac{12EI_z}{L^3} & 0 & 0 & 0 & -\frac{6EI_z}{L^2} \\ 0 & 0 & \frac{12EI_y}{L^3} & 0 & \frac{6EI_y}{L^2} & 0 \\ 0 & 0 & 0 & \frac{GJ}{L} & 0 & 0 \\ 0 & 0 & \frac{6EI_y}{L^2} & 0 & \frac{4EI_y}{L} & 0 \\ 0 & -\frac{6EI_z}{L^2} & 0 & 0 & 0 & \frac{4EI_z}{L} \end{bmatrix}$$

Ensamblando, obtenemos la matriz de rigidez de un elemento de pórtico espacial en coordenadas locales:

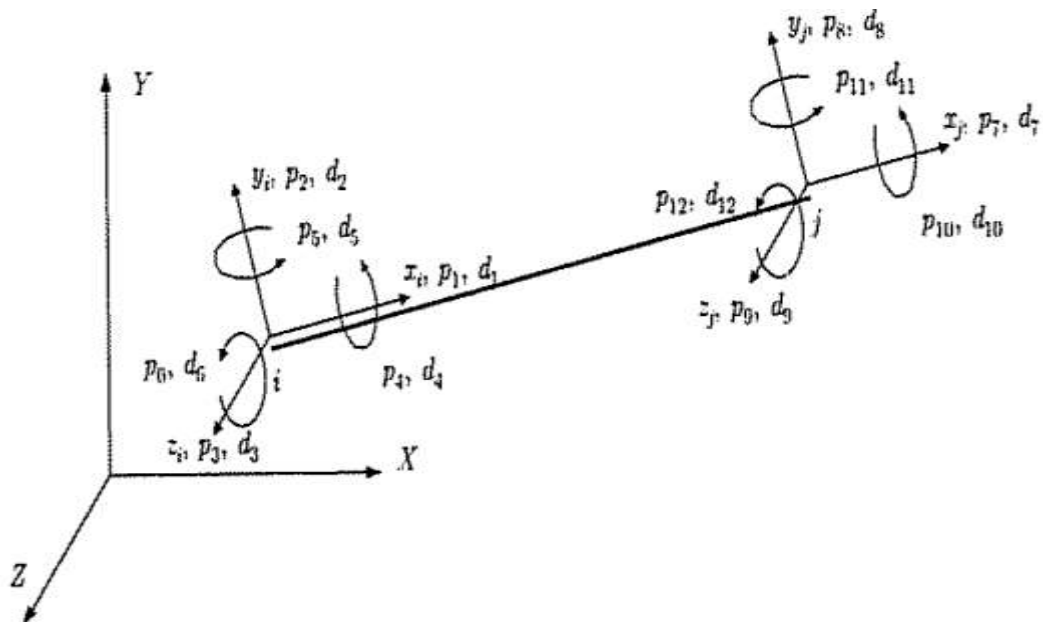
$$\bar{K} = \begin{bmatrix} \frac{AE}{L} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \frac{12EI_z}{L^3} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{12EI_y}{L^3} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{GJ}{L} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -\frac{6EI_y}{L^2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \frac{6EI_z}{L^2} & 0 & 0 & 0 & -\frac{6EI_z}{L^2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -\frac{AE}{L} & 0 & 0 & 0 & 0 & 0 & \frac{AE}{L} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \frac{6EI_z}{L^2} & 0 & 0 & 0 & 0 & 0 & \frac{12EI_z}{L^3} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{6EI_y}{L^2} & 0 & 0 & 0 & 0 & 0 & \frac{12EI_y}{L^3} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{GJ}{L} & 0 & 0 & 0 & 0 & 0 & \frac{GJ}{L} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{4EI_y}{L} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \frac{6EI_z}{L} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \frac{AE}{L} & 0 & 0 & 0 & 0 & 0 & -\frac{AE}{L} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \frac{6EI_z}{L^2} & 0 & 0 & 0 & 0 & 0 & \frac{12EI_z}{L^3} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -\frac{6EI_y}{L^2} & 0 & 0 & 0 & 0 & 0 & \frac{12EI_y}{L^3} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -\frac{GJ}{L} & 0 & 0 & 0 & 0 & 0 & -\frac{GJ}{L} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{2EI_y}{L} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \frac{6EI_z}{L^2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{6EI_z}{L^2} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{4EI_z}{L} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

2.4.- Tipos de Estructuras.-

2.4.1.- Estructuras espaciales.- Son aquellas en las que tanto los elementos estructurales como las cargas aplicadas pueden ubicarse y orientarse libremente en el espacio. Se consideran los siguientes tipos: **Pórticos espaciales**, celosías espaciales y emparrillados.

2.4.1.1.-Pórticos espaciales (pórticos en el espacio).- Se trata de estructuras cuyos elementos y cargas están ubicados espacialmente con toda libertad. Para cada nudo se consideran SEIS grados de libertad (tres traslaciones y tres giros). Se trata del tipo estructural más genérico.

Ilustración 21. Sistemas de coordenadas global y local para elementos espaciales de nudos rígidos.



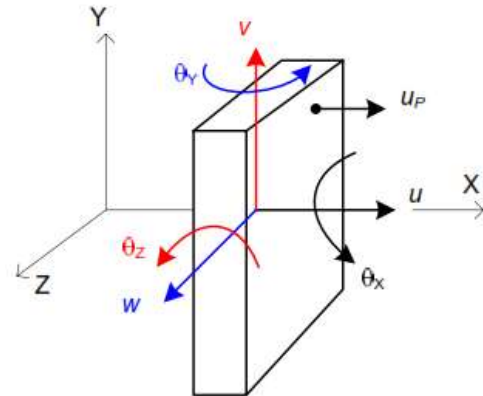
Los elementos utilizables para configurar estos sistemas son: barras y elementos superficiales tipo lamina y sólidos.

2.4.1.2. - Barra de un pórtico tridimensional.-

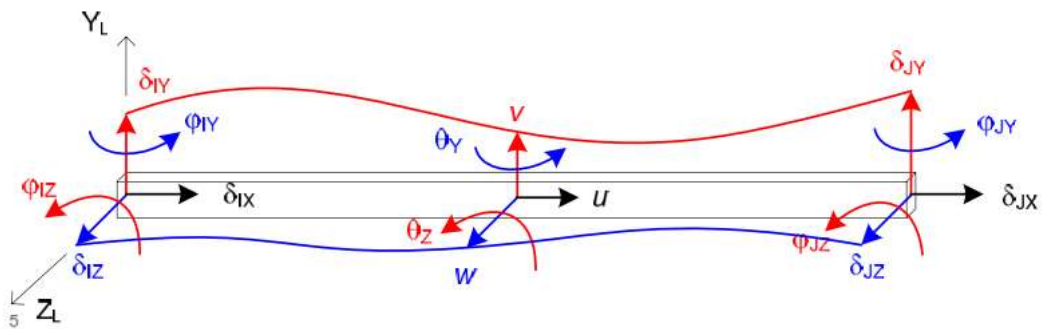
Deformaciones de la fibra neutra:

Axial u , laterales v , w , giros según X, Y, Z

Deformaciones de un punto P fuera de la fibra neutra:

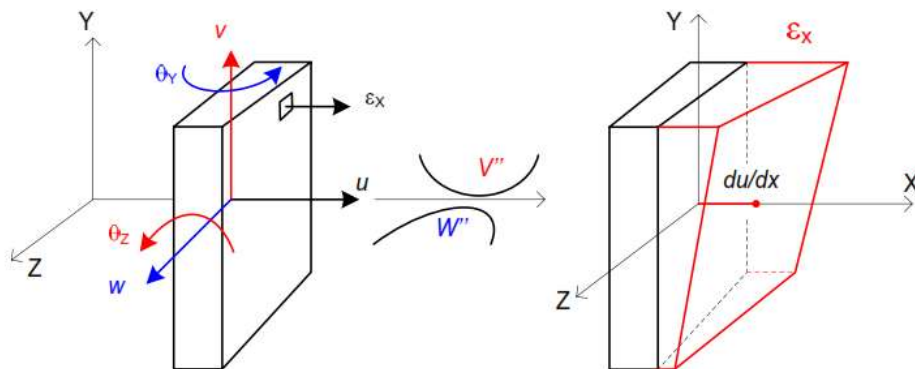


$$u_p = u - \theta_z y + \theta_y z = u - \frac{dv}{dx} y - \frac{dw}{dx} z$$



✚ Deformación unitaria axial debida a la flexión y axial:

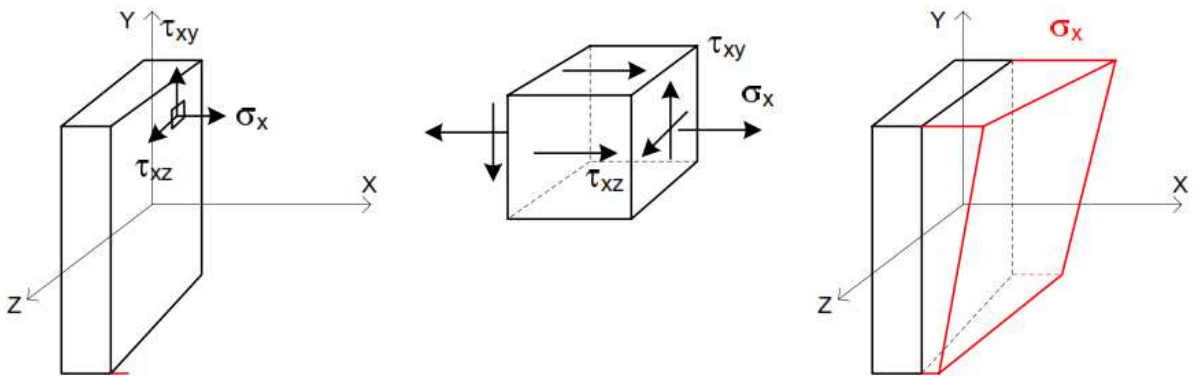
$$\epsilon_x = \frac{\partial u_p}{\partial x} = \frac{du}{dx} - \frac{d^2v}{dx^2} y - \frac{d^2w}{dx^2} z$$



Ecuación constitutiva lineal:

$$\sigma_x = E(\varepsilon_x - \alpha T)$$

$$\sigma_x = E(u' - v''y - w''z - \alpha T)$$

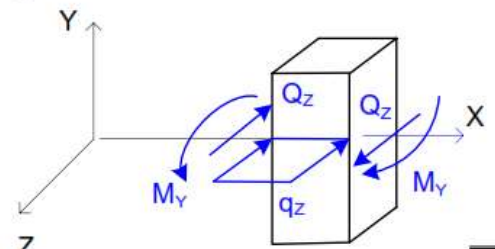
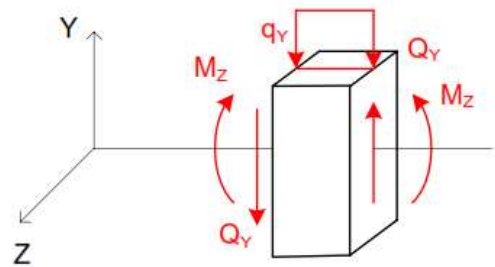
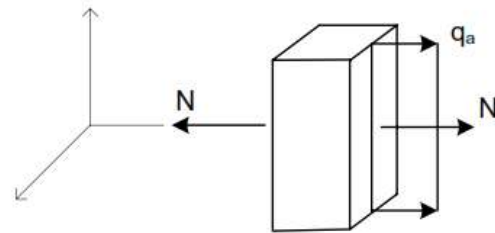


✚ Esfuerzos en un barra tridimensional.-

$$N \equiv \int \sigma dA = EAu' - EA\alpha T_m$$

$$M_z \equiv -\int \sigma y dA = EI_z v'' + EI_z \alpha T_m$$

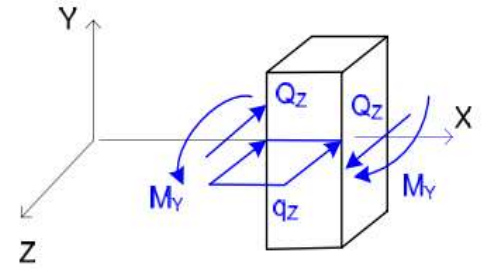
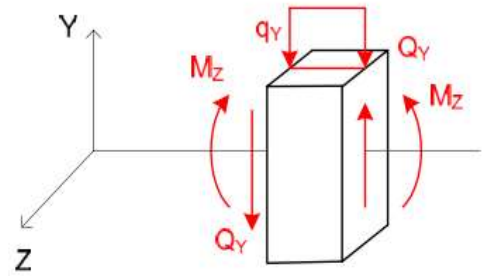
$$M_y \equiv -\int \sigma z dA = EI_y w'' + EI_y \alpha T_m$$



✚ Cortantes:

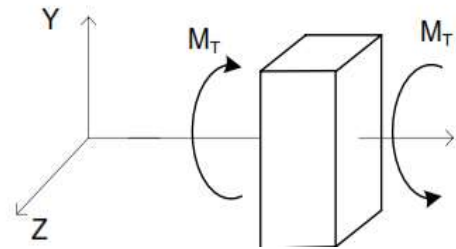
$$Q_Y = \int \tau_{xy} dA$$

$$Q_Z = \int \tau_{xz} dA$$

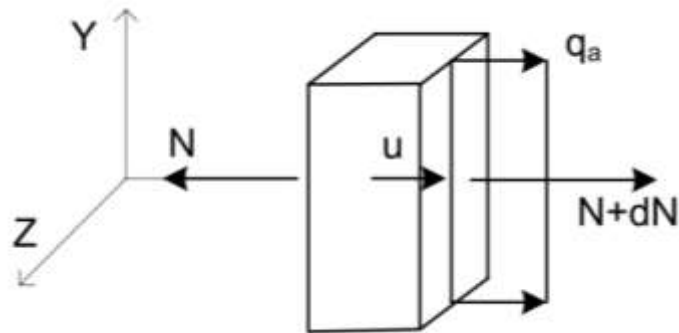


✚ Momento Torsor:

$$M_T = \int (\tau_{xz} y - \tau_{xy} z) dA$$



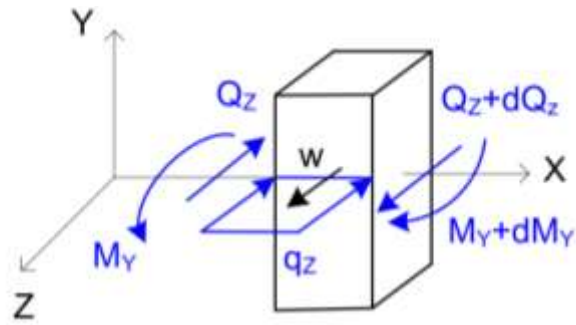
✚ Barra a flexión en el espacio.- Ecuaciones de equilibrio.-



Fuerza Axial:

$$q_a = EA \frac{d^2 u}{dx^2}$$

Propiedades Uniformes:

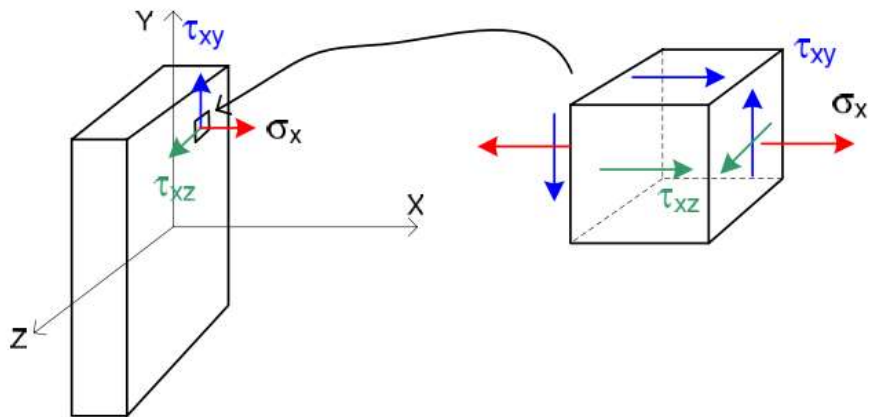


Momentos s/ Y $Q_z = -\frac{dM_Y}{dx}$

Fuerzas s/ Z $q_z = -EI_Y \frac{d^4 w}{dx^4}$

Flexión y esfuerzo axial:

$$\sigma_x = \frac{N}{A} - \frac{M_z y}{I_z} - \frac{M_y z}{I_y}$$



Esfuerzos cortantes:

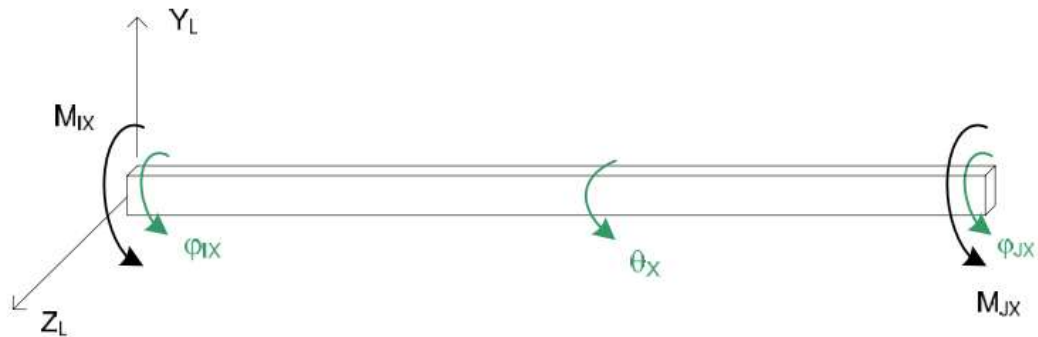
$$\tau_{XY} = \frac{Q_Y \bar{A}_Z}{I_Z b_Z} \quad \tau_{XZ} = \frac{Q_Z \bar{A}_Y}{I_Y b_Y}$$

✚ Barra en el espacio.- Energía.-

Energía acumulada en toda la barra (sin energía de cortante ni torsión):

$$\begin{aligned}
 U_b^* &= \int \frac{N^2}{2EA} dx + \int N\alpha T_m dx \\
 &+ \int \frac{M_z^2}{2EI_z} dx - \int M_z\alpha T_{\theta z} dx \\
 &+ \int \frac{M_y^2}{2EI_y} dx - \int M_y\alpha T_{\theta y} dx
 \end{aligned}$$

Torsión.-



$$M_{IX} = \frac{GJ}{L}(\varphi_{IX} - \varphi_{JX})$$

$$M_{JX} = -M_{IX}$$

$$U_{Tb} = \int \frac{M_T^2}{2GJ} dx$$

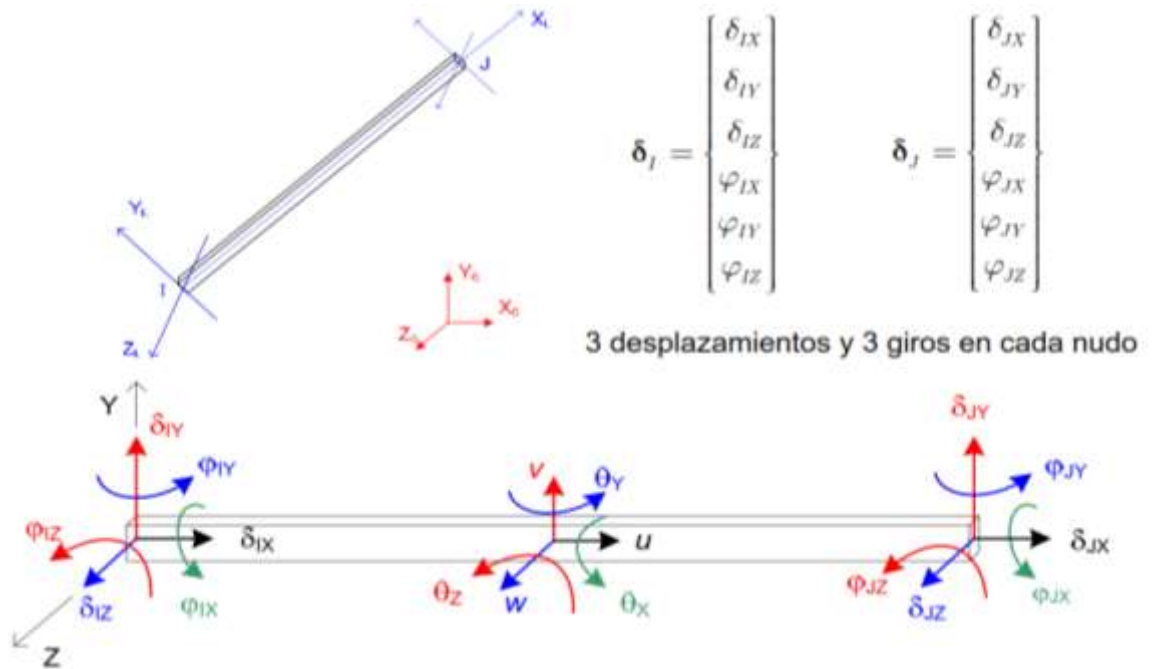
Rigidez a la torsión: GJ/L

G: módulo de elasticidad en cortadura

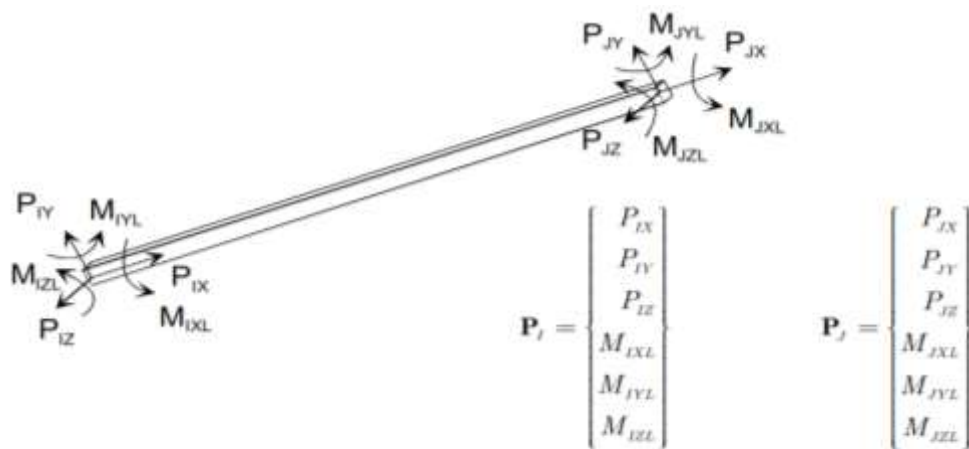
Sección circular: J = momento de inercia polar

Otras secciones: J según la teoría de la torsión

Grados de Libertad.-



Fuerzas en los nudos.-



3 fuerzas y 3 momentos en cada nudo

✚ Rigidez en el Sistema Local.-

Matrix de 12x12

4 sub matrices de 6x6

4 efectos desacoplados:

2 flexiones (XY, YZ), Axial Torsión

Se obtiene ensamblando las matrices de:

Viga plana en XY (4 gdl).

Viga Plana en XZ (4 gdl).

Barra Axial (2 gdl).

Barra a torsión (2 gdl).

$$\begin{bmatrix} P_{IX} \\ P_{IY} \\ P_{IZ} \\ M_{IXL} \\ M_{IYL} \\ M_{IZL} \\ \hline P_{JX} \\ P_{JY} \\ P_{JZ} \\ M_{JXL} \\ M_{JYL} \\ M_{JZL} \end{bmatrix} = \begin{bmatrix} \mathbf{K}_{LII} & \mathbf{K}_{LIJ} \\ \mathbf{K}_{LJI} & \mathbf{K}_{LJJ} \end{bmatrix} \begin{bmatrix} \delta_{IX} \\ \delta_{IY} \\ \delta_{IZ} \\ \varphi_{IX} \\ \varphi_{IY} \\ \varphi_{IZ} \\ \hline \delta_{JX} \\ \delta_{JY} \\ \delta_{JZ} \\ \varphi_{JX} \\ \varphi_{JY} \\ \varphi_{JZ} \end{bmatrix}$$

$$\mathbf{K}_{LIT} = \begin{bmatrix} \frac{EA}{L} & 0 & 0 & 0 & 0 & 0 \\ 0 & \frac{12EI_z}{L^3} & 0 & 0 & 0 & \frac{6EI_z}{L^2} \\ 0 & 0 & \frac{12EI_y}{L^3} & 0 & -\frac{6EI_y}{L^2} & 0 \\ 0 & 0 & 0 & \frac{GJ}{L} & 0 & 0 \\ 0 & 0 & -\frac{6EI_y}{L^2} & 0 & \frac{4EI_y}{L} & 0 \\ 0 & \frac{6EI_z}{L^2} & 0 & 0 & 0 & \frac{4EI_z}{L} \end{bmatrix}$$

4 efectos desacoplados:
2 flexiones (XY, XZ)
axial (X)

2.4.2.- El análisis de estructuras por el M.E.F 2D y 3D.-

El análisis estático de estructuras laminares por el M.E.F. puede abordarse desde dos puntos de vista diferentes, uno 2D y el otro 3D. El primero de ellos remarca el carácter bidimensional de estas piezas, considerando el comportamiento global en el espesor y requiriendo para ello la consideración de ecuaciones constitutivas especiales, pues, una simplificación en cuanto a la cinemática y, en general, también en lo relativo a las ecuaciones constitutivas, así como un tiempo de computación necesario bastante menor.

Por el contrario, el planteamiento 3D se basa en la determinación precisa de la respuesta en cada punto, utilizando las relaciones constitutivas convencionales y sin requerir, en principio, ninguna simplificación ni cinemática ni constitutiva; en contrapartida, puede presentar problemas numéricos, debido a la gran diferencia de rigidez en unas u otras direcciones, y requiere mayor tiempo de computación.

Desde el punto de vista de la cinemática, el enfoque 3D parece, a primera vista, notablemente superior, porque no requiere simplificación alguna. No obstante, esto no es así, porque considerando un solo elemento en espesor, como es habitual, la función de interpolación en esta dirección jugará un papel similar al de la hipótesis cinemática en un modelo 2D, y la consideración de varios elementos en ella agravará los problemas

de mal condicionamiento numérico y elevado coste computacional. Así pues, en este aspecto deben considerarse superiores los modelos 2D.

En cuanto a las ecuaciones constitutivas, en el ámbito de los materiales es posible desarrollar una versión 2D de ellas, es decir, una relación entre esfuerzos generalizados y deformaciones generalizadas, a partir de la 3D y equivalente a aquella, por lo que se puede realizar un planteamiento global del problema completamente 2D equivalente al 3D de partida. Sin embargo, cuando se trata de análisis elastoplásticos, no es posible, en el marco de la teoría del flujo plástico, obtener una versión 2D de las ecuaciones constitutivas equivalente a la 3D inicial, debido, por una parte, a que la consideración de partes plásticas de las deformaciones generalizadas implica un modo de extenderse la plastificación en el espesor diferente del propio del análisis 3D, y por otra, a que no puede determinarse una superficie de plastificación en esfuerzos generalizados (o deformaciones generalizadas) partiendo de una dada en tensiones 3D (o deformaciones 3D). En consecuencia, en el análisis elastoplástico, deberá optarse por una formulación 3D para poder introducir las características del material en forma convencional, pudiendo incorporar así, sin mayor complicación, todos los avances que en este campo se produzcan a medida que ello suceda, o bien recurrir a una teoría del flujo plástico de las láminas postulada a priori, sobre la base de las simplificaciones necesarias, y sujeta a verificación por contraste con modelos 3D o por métodos experimentales. En esta última línea, justificada por el menor coste computacional, cabe citar los modelos desarrollados por Ilyushin, Ivano, Cristfiels, Eidsheim, Larzen y Bienik y Funaro, a todos los cuales es común considerar comportamiento perfectamente plástico, despreciar la influencia de los cortantes y partir del criterio 3D de Hubert-von Mises-Hencky.

En vista, pues, de que para mantener la generalidad suficiente en cuanto al modelo de comportamiento del material, y no introducir formas aberrantes de extensión de la plastificación por el espesor en él, se hace necesario recurrir a una formulación 3D. Procede abordar a continuación el comentario de sus inconvenientes y de los medios que se proponen para eludirlos.

Es bien conocido que el análisis de estructuras marcadamente bidimensionales mediante elementos finitos 3D convencionales presenta importantes problemas numéricos. Para solucionarlos diferentes autores han utilizado procedimientos distintos, entre los cuales el que mayor éxito ha sido el de definir el campo de desplazamientos del espacio laminar a partir exclusivamente de parámetros significativos en el movimiento de la lámina considerada como estructura 2D, es decir, de los que definen el movimiento de su superficie media y la evolución de la normal. Tal método, en esencia supone admitir una hipótesis de comportamiento transversal de la lámina, y con diferentes expresiones de ésta ha sido utilizado por varios. En este escrito se presenta uno de estos elementos (entendiendo como tal manera la descripción de unas funciones de interpolación, sino el procedimiento completo de obtención de las matrices cinemáticas que aparecen en el M.E.F.), basado en el modelo de comportamiento transversal, el cual se ha desarrollado siguiendo un método que permite aumentar el número de puntos de integración en el espesor considerablemente con muy poco coste computacional adicional, por lo que se considera muy adecuado para el análisis.

En un trabajo anterior, los autores presentaban un procedimiento de obtención de las matrices cinemáticas que intervienen en el problema estático de una lámina planteado por el M.E.F. basado en la descomposición del proceso en dos bloques: el primero de ellos permite obtener las citadas matrices cinemáticas en función de las primeras y segundas derivadas de Frechet del gradiente del desplazamiento en la dirección de los incrementos de los parámetros nodales, y es común a cualquier problema estático de un continuo 3D; el segundo se encarga de calcular dichas derivadas de Frechet, siendo el único que tiene en cuenta el carácter laminar de la estructura. Este planteamiento presenta la ventaja de ser común el primer bloque para cualquier problema estático 3D, y ser el segundo muy sencillo y basado exclusivamente en conceptos elementales de análisis tensorial y geometría diferencial, por lo que resulta fácil modificarlo para considerar otras hipótesis cinemáticas; por el contrario, presenta el inconveniente de exigir un tiempo de computación elevado cuando se procesa una rutina, puesto que el número de operaciones a realizar en cada punto de integración en el espesor es elevado.

En el presente escrito, partiendo de un planteamiento similar, se combinarán ambos bloques para generar dos nuevos, el primero, con un número importante de operaciones, se ejecuta una sola vez por punto de integración considerado sobre la superficie de referencia, y el segundo, muy simple y rápido de ejecución, consiste en una combinación polinómica de valores previamente hallados y es el único que debe calcularse en todos y cada uno de los puntos de integración del espesor. Con este nuevo planteamiento se ha perdido la posibilidad de fácil modificación del anterior, así como la entidad conceptual de cada uno de los bloques, pero a cambio se ha conseguido mayor velocidad de proceso y la posibilidad de identificar los distintos términos de las matrices A y B con las primeras y segundas variaciones, respectivamente, de las deformaciones generalizadas que aparecen en la formulación 2D.

2.4.3.- Identificación estructural en 2D.-

En esta primera parte se definirá la estructura a base de datos y números.

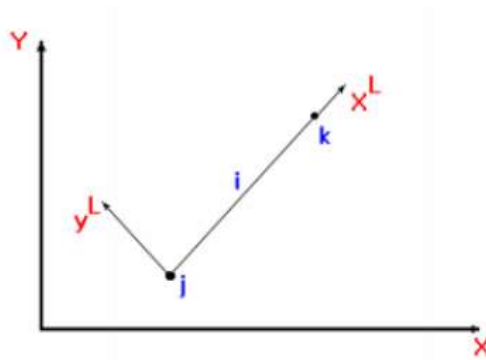
- ✚ Conectividad de los elementos, se identifica para cada barra el nodo inicial y final. La misma queda definida automáticamente por el orden establecido para la numeración de los nodos de la barra.

- ✚ El eje 'x' local, coincide con el eje geométrico de la barra, siendo el sentido positivo el que va del nodo de menor numeración al de mayor numeración. Los otros ejes formarán un triedro directo.

2.4.3.1.-Matriz de rigidez en el plano.-

- **Estructura articulada:** Consideramos la barra de una estructura articulada con ejes locales x' e y' orientados de la manera que comentábamos anteriormente.

Supondremos que tratamos con una barra recta de sección transversal constante que responde a la Ley de Hooke.



Ejes locales y globales de una barra

En la barra i de la figura 3, el nodo inicial es el 'j' y el final es el 'k', quedando definida la orientación de la misma por los ejes locales 'x' e 'y'.

Se considera que no existen deformaciones iniciales y que la deformación es elástica. En este caso el alargamiento de la barra 'i' estará dado por:

$$\Delta L_i = X_{X_k}^L - X_{X_j}^L$$

Donde son los desplazamientos producidos en la barra en dirección del eje x.

En el caso de estructuras articuladas, la única sollicitación que podremos encontrar será el esfuerzo axial, por lo que, siendo el nudo 'j' el inicial y 'k' el final tendremos:

$$F_{X_j} = \frac{E * A}{L} \Delta L_i = -\frac{E * A}{L} (X_{X_k}^L - X_{X_j}^L)$$

$$F_{X_k} = \frac{E * A}{L} \Delta L_i = \frac{E * A}{L} (X_{X_k}^L - X_{X_j}^L)$$

Siendo:

E , El módulo de elasticidad, A el área transversal y L la longitud de la barra.

Sabiendo que en el eje y' local de cada barra no podemos encontrar ningún esfuerzo, podremos expresar de forma matricial las ecuaciones anteriores:

En esta ecuación vienen definidas las fuerzas extremo de ambos nodos, el inicial y el final de la barra para cualquier pareja de desplazamientos. Estas ecuaciones son simétricas, como se podía esperar a partir del teorema de reciprocidad. No es posible, sin embargo, resolverlas y obtener los desplazamientos (X) en términos de las fuerzas (F), puesto que la matriz K es singular. Esta información nos lleva a concluir que las barras pueden estar sometidas a cualquier esfuerzo arbitrario que no afectará a los movimientos del nodo inicial y final de la barra.

En el caso de que la estructura esté dispuesta de un muelle sometido a tracción

$$F_{X_k} = K * \Delta L_i = K * (X_{X_k}^L - X_{X_j}^L)$$

$$F_{Y_k} = K * \Delta L_i = K * (X_{X_k}^L - X_{X_j}^L)$$

Donde K es la constante de rigidez del muelle.

$$\begin{bmatrix} F_{X_j} \\ F_{Y_j} \\ F_{X_k} \\ F_{Y_k} \end{bmatrix} = \begin{bmatrix} K & 0 & K & 0 \\ 0 & 0 & 0 & 0 \\ K & 0 & K & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} * \begin{bmatrix} X_{X_j}^L \\ X_{Y_j}^L \\ X_{X_k}^L \\ X_{Y_k}^L \end{bmatrix}$$

Mientras que si nos encontramos con un muelle que haga de efecto disipador tendremos para los nudos 'j' y 'k'.

$$F_{X_k} = C * \Delta L_i = C * (\dot{X}_{X_k}^L - \dot{X}_{X_j}^L)$$

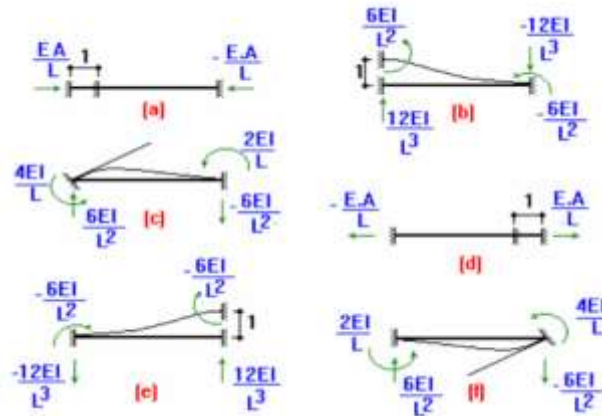
$$F_{Y_k} = C * \Delta L_i = C * (\dot{X}_{X_k}^L - \dot{X}_{X_j}^L)$$

Siendo C la constante de amortiguamiento del amortiguador.

$$\begin{bmatrix} F_{X_j} \\ F_{Y_j} \\ F_{X_k} \\ F_{Y_k} \end{bmatrix} = \begin{bmatrix} C & 0 & C & 0 \\ 0 & 0 & 0 & 0 \\ C & 0 & C & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} * \begin{bmatrix} \dot{X}_{X_j}^L \\ \dot{X}_{Y_j}^L \\ \dot{X}_{X_k}^L \\ \dot{X}_{Y_k}^L \end{bmatrix}$$

- **Estructura reticulada:** En este tipo de estructura nos encontramos con los movimientos traslacionales en el eje horizontal y vertical más el giro en el plano.

Para la obtención de la matriz de rigidez, se aplican desplazamientos de valor unitario en uno de los movimientos disponibles y se restringen los demás.



Reacciones unitarias

Las reacciones que se hallan con los desplazamientos unitarios serán los términos que formarán la matriz de rigidez de la estructura:

$$\begin{bmatrix} F_{Xj} \\ F_{Yj} \\ F_{Zj} \\ F_{Xk} \\ F_{Yk} \\ F_{Zk} \end{bmatrix} = \begin{bmatrix} \frac{EA}{L} & 0 & 0 & -\frac{EA}{L} & 0 & 0 \\ 0 & \frac{12EI}{L^3} & \frac{6EI}{L^2} & 0 & -\frac{12EI}{L^3} & \frac{6EI}{L^2} \\ 0 & \frac{6EI}{L^2} & \frac{4EI}{L} & 0 & -\frac{6EI}{L^2} & \frac{4EI}{L} \\ -\frac{EA}{L} & 0 & 0 & \frac{EA}{L} & 0 & 0 \\ 0 & -\frac{12EI}{L^3} & -\frac{6EI}{L^2} & 0 & \frac{12EI}{L^3} & -\frac{6EI}{L^2} \\ 0 & \frac{6EI}{L^2} & \frac{4EI}{L} & 0 & -\frac{6EI}{L^2} & \frac{4EI}{L} \end{bmatrix} * \begin{bmatrix} X_{Xj}^L \\ X_{Yj}^L \\ X_{Zj}^L \\ X_{Xk}^L \\ X_{Yk}^L \\ X_{Zk}^L \end{bmatrix}$$

De la misma, expresada de forma compacta:

$$F I^L = K I^L * X I^L$$

2.4.4.- Método de elementos finitos variacional.-

La formulación de elementos finitos puede deducirse para ciertos problemas, como por ejemplo el análisis de estructuras, como una extensión de los métodos matriciales utilizados para calcular estructuras de vigas y reticulados. Sin embargo, dicha deducción encuentra serias limitaciones cuando se quiere extender la formulación a problemas no estructurales. Por ello se mostrarán en este apunte algunos conceptos básicos de la formulación variacional del método de elementos finitos que pueden aplicarse a una gran variedad de problemas. En primer lugar describiremos algunos conceptos sobre métodos aproximados de solución para ecuaciones diferenciales, en particular veremos el método de Raleigh-Ritz y veremos la utilización de este método con los elementos finitos y se describirá la implementación matricial.

✚ **Métodos aproximados de solución.-** Describiremos el método más usual para la resolución aproximada de ecuaciones diferenciales en dos y tres dimensiones, sobre los que basan la mayoría de implementaciones de elementos finitos. En general los métodos empleados son de dos tipos, por un lado están los métodos basados en principios variacionales, generalmente asociados a la minimización de algún funcional, y por otro lado tenemos los métodos del tipo de residuos ponderados que se aplican directamente sobre la ecuación diferencial y sus condiciones de contorno, no precisando de ningún funcional asociado.

✚ **Método De Rayleigh-Ritz** Con este método es posible obtener soluciones aproximadas de ecuaciones diferenciales mediante principios variacionales. La idea básica consiste en aproximar a las soluciones u, v que hacen estacionario un funcional mediante una suma ponderada de funciones

$$\tilde{u} = \sum_{i=1}^m a_i N_i(x, y), \quad \tilde{v} = \sum_{i=m+1}^n a_i N_i(x, y)$$

donde a_i son constantes a determinar llamadas coordenadas generalizadas. Las funciones $N_i(x, y)$ son llamadas funciones de prueba y pueden ser elegidas arbitrariamente.

Consideremos un sistema de ecuaciones diferenciales cuya solución es equivalente a hacer

estacionaria la primera variación de un funcional asociado, Π que es función de u , v y sus derivadas primeras:

$$\Pi = \int_{\Omega} F(x, y, u, v, u_x, u_y, v_x, v_y) dx dy$$

Las derivadas de las funciones de aproximación u son:

$$\frac{\partial \tilde{u}}{\partial x} = \sum_{i=1}^m a_i \frac{\partial N_i(x, y)}{\partial x}, \quad \frac{\partial \tilde{u}}{\partial y} = \sum_{i=1}^m a_i \frac{\partial N_i(x, y)}{\partial y}$$

y en forma análoga para la aproximación de v . Si sustituimos las funciones de aproximación u , v y sus derivadas en el funcional Π este se transformará en una función de las coordenadas generalizadas a_i , cuyos valores por ahora desconocemos, esto es:

$$\Pi = \Pi(a_i) \quad i = 1, 2, \dots, n$$

Luego deseamos conocer cuales son los mejores valores de las constantes a_i , tal que reemplazados en las expresiones nos brinden la mejor aproximación a la solución del sistema de ecuaciones diferenciales asociado al funcional Π . Para ello aplicamos la condición de estacionaridad a este funcional, que sabemos que debe ser satisfecha por la solución exacta, resultando:

$$\delta \Pi = \sum_{i=1}^n \frac{\partial \Pi}{\partial a_i} \delta a_i = 0$$

Como esta ecuación debe ser válida para variaciones arbitrarias δa_i , luego deben cumplirse las siguientes n ecuaciones algebraicas:

$$\frac{\partial \Pi}{\partial a_i} = 0 \quad i = 1, 2, \dots, n$$

Si ahora consideramos el caso particular, pero muy común en problemas físicos, donde el funcional Π es una función cuadrática de las funciones u , v y sus derivadas primeras, entonces al substituir las aproximaciones dadas por la ecuación este funcional será una

función cuadrática de las coordenadas generalizadas a_i . Por lo tanto, las derivadas de este funcional serán funciones lineales en las coordenadas generalizadas a_i que se pueden expresar como:






$$\frac{\partial \Pi}{\partial a_i} = (k_{i1}a_1 + k_{i2}a_2 + \dots + k_{in}a_n - f_i) = 0 \quad i = 1, 2, \dots, n$$

2.5.- Lenguaje de programación.-

2.5.1.-MatLab.- Es una abreviatura de la frase Matrix Laboratory. Es un entorno informático de análisis numérico y representación gráfica de fácil manejo. Originalmente fue escrito para la enseñanza de álgebra lineal, aunque actualmente es, al mismo tiempo, un entorno y un lenguaje de programación. También permite crear funciones propias y programas especiales (denominados archivos-M) en código MatLab, que se pueden agrupar en las llamadas Toolboxes: colección especializada de archivos-M para trabajar en distintos tipos de problemas, por ejemplo de optimización, de estadística, de ecuaciones diferenciales parciales, etc.

Se puede considerar, por otro lado, que MatLab es una calculadora totalmente equipada aunque, en realidad, es mucho más versátil que cualquier calculadora para hacer cálculos matemáticos. Se trata de una plataforma para el desarrollo de aplicaciones y para la resolución de problemas en múltiples áreas de aplicación.

Entre sus utilidades, se encuentra:

-  Cálculo matricial y Algebra lineal.
-  Polinomios e interpolación.
-  Regresión y ajuste de funciones.
-  Ecuaciones diferenciales ordinarias.
-  Integración.

2.5.1.1.- Simbología.-

Los símbolos A y a son diferentes para MatLab: se distingue entre mayúsculas y minúsculas.

Se pueden escribir comentarios después del signo de tanto por ciento (%).

Podemos colocar órdenes múltiples en una línea si se separan por comas o puntos y comas.

- ✚ Las comas le dicen a MatLab que visualice los resultados. Los puntos y comas suprimen la impresión.
- ✚ Para separar una línea en varias se ponen puntos suspensivos: ...
- ✚ Para interrumpir la ejecución de una instrucción o programa de MatLab en cualquier momento: Ctrl-C

2.5.2.- Funciones matemáticas comunes.-

A continuación se muestra una tabla con las funciones matemáticas en MatLab:

`abs(x)` Valor absoluto o magnitud de un número complejo

`acos(x)` Inversa del coseno

`acosh(x)` Inversa del coseno hiperbólico `angle(x)` Angulo de un número complejo `asin(x)` Inversa del seno

`asinh(x)` Inversa del seno hiperbólico

`atan(x)` Inversa de la tangente

`atan2(x,y)` Inversa de la tangente en los cuatro cuadrante

`atanh(x)` Inversa de la tangente hiperbólica `ceil(x)` Redondea hacia más infinito `conj(x)` Complejo conjugado

`cos(x)` Coseno

`cosh(x)` Coseno hiperbólico

`exp(x)` Exponencial

`fix(x)` Redondea hacia cero

`floor(x)` Redondea hacia menos infinito

`imag(x)` Parte imaginaria de un número complejo

`log(x)` Logaritmo natural

`log10(x)` Logaritmo decimal

`real(x)` Parte real de un número complejo

`rem(x,y)` Resto después de la división

`round(x)` Redondea hacia el entero más próximo

`sign(x)` Devuelve el signo del argumento

`sin(x)` Seno

`sinh(x)` Seno hiperbólico `sqrt(x)` Raíz cuadrada `tan(x)` Tangente

`tanh(x)` Tangente hiperbólica

2.5.2.1.-Notas:

- ✚ MatLab sólo opera en radianes.
- ✚ Para ver las diferentes funciones elementales y trigonométricas teclear `help elfun`
- ✚ La siguiente orden borra de memoria todas las variables: `clear`

2.5.3.-Archivos M

Se pueden colocar órdenes en un simple archivo de texto (o ascii) y, a continuación, hacer que MatLab lo abra y evalúe las órdenes exactamente como si hubiesen sido escritas desde la línea de comandos.

Estos archivos se llaman archivos script o archivos-M, y deben finalizar con la extensión 'm'.

Para crear un archivo-M se escoge New del menú File y seleccionamos M-file. Una vez guardado este archivo-M, MatLab ejecutará las órdenes en dicho archivo simplemente escribiendo su nombre (sin extensión) en la línea de comandos.

Normalmente, las órdenes leídas desde el archivo-M no se visualizan cuando se evalúan. La orden `echo on` le dice a MatLab que visualice o efectúe un eco de las órdenes en la ventana de Orden cuando se leen y evalúan. También existe la función `echo off`.

254-Funciones elementales para la construcción de matrices.-

`zeros(n)` Matriz de ceros ($n \times n$)

`ones(n,m)` Matriz de unos ($n \times m$).

`rand(n,m)` Matriz ($n \times m$) de números aleatorios distribuidos uniformemente entre cero y uno.

`randn(n,m)` Matriz ($n \times m$) de números aleatorios distribuidos normalmente con media cero y varianza unidad.

2.5.4.1.-Utilización del carácter :

Este comando es de mucha utilidad a la hora de programar con vectores y matrices. Con

el carácter `:` Se pueden extraer partes de una matriz. Los siguientes ejemplos explican su utilización

`x = [0:0.1:2]`

vector de 0 a 2 con paso 0.1

`I = eye(6)`

matriz identidad 6x6

`x = I(1,2)` elemento (1,2) de I `x = I(3,:)`

toda la fila tercera de I

`x = I(:,2)`

toda la columna segunda de I

$B = I(2:5,2:6)$

filas 2 a 5 y columnas 2 a 6

$I(:, [2\ 4\ 6]) = R(:, 1:3)$

Reemplaza las columnas 2,4,6 de I por las 3 primeras de R

También son útiles los siguientes comandos:

$v = R(:)$

vector que contiene a todas las columnas de R

$Ru = \text{triu}(R)$

matriz triangular superior de R

$Rl = \text{tril}(R)$

matriz triangular inferior de R

2.5.5.-Guardar resultados.-

En MatLab existen dos modalidades para guardar resultados. La primera es mediante la instrucción,

`diary('fichero1'`, esta orden, poniéndola al principio de la sesión, guarda en el fichero de texto fichero1 todo lo que se haya realizado en una sesión de MatLab. Esta orden sólo guarda los comandos introducidos en la línea de comandos pero no las variables y matrices. Para guardarlas y cargarlas se teclaa

`save fichero2`

guarda todas las variables en fichero2.mat

Este formato sólo es legible mediante MatLab

`save fichero3 x A`

guarda sólo las variables x y A en fichero3.mat

`save fichero4.dat x A -ascii`

guarda las variables x y A en ASCII en fichero4.dat

`load fichero3`

carga variables de fichero3.mat (binario)

`load fichero4.dat -ascii`

carga variables de fichero4.dat (ASCII) sea cual sea su extensión.

`who`

Lista todas las variables del espacio de trabajo.

`clear`

Borra todas las variables del espacio de trabajo.

`clear(v1,v2)`

Borra todas las variables v1 y v2 del espacio de trabajo.

2.5.6.-Gráficos.-

MatLab presenta un entorno gráfico de muy fácil manejo. El ejemplo más sencillo para

crear gráficos es el siguiente

```
x=rand(10,1); y = rand(10,1); plot(x,y)
```

El comando `plot` dibuja los puntos $(x_i; y_i)$ uniéndolos por líneas continuas. Para dibujar un diagrama de dispersión de las variables x e y se tecllea

```
grid on title('grafico') text(1,0.65,'y = x log(x)')
```

Nota: el comando `grid on` imprime una malla en el gráfico.

Ejemplo: para dibujar la función $y = x \sin(1/x)$ se escribe

```
plot(x,x.*sin(ones(size(x))./x))
```

El comando `size` obtiene las dimensiones del argumento. Así, con la orden `ones` se dibuja una vector columna de unos de tamaño `size` de `x` (esto es la longitud del vector `x`) y se divide luego entre el valor de cada `x`.

Observar el uso del comando `ones` y del `./`

Existe una función que evalúa cuidadosamente la función que se va a representar. Como entrada, esta función necesita conocer el nombre de la función en forma de una cadena de caracteres y el rango de representación como un array de dos elementos: `fplot('nombre',[a,b])`

```
fplot('t*sin(1/t)',[0 3])
```

Para definir los ejes X e Y de un gráfico se utiliza la orden `axis([Xmin Xmax Ymin Ymax])`

Para gráficos en tridimensional se pone:

```
plot3(x,y,z)
```

2.5.7.-Controles de flujo.-

```
for x = array ordenes
```

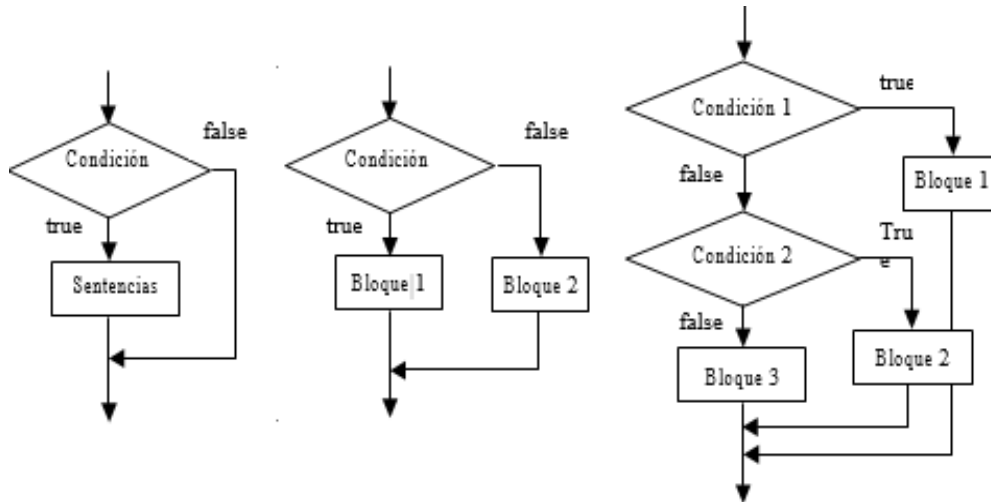
```
while expresion ordene
```

Un bloque `while` que ejecuta órdenes mientras todos los elementos de `expresion` son verdaderas o distinto de cero.

```
if expresion ordenes
```

```
end
```

Una simple estructura `if-else-end` donde `ordenes` se ejecutan si todos los elementos en expresión son verdaderas



if expresion

ordenes evaluadas si exp=verdadero els

ordenes evaluadas si exp=falso

end

Una estructura if-else-end con dos caminos. Un grupo de órdenes se ejecuta si

expresion es verdadera. El otro conjunto se ejecuta si expresion es falsa o cero.

if expresion1

ordenes evaluadas si expresion1 es verdadera elseif expresion2

ordenes evaluadas si expresion2 es verdadera elseif

... else

ordenes evaluadas si ninguna otra expresion es verdadera end

Es la estructura más general **if-else** end.

break Termina la ejecución de bucles for y bucles while.

Notas:

La condición del if puede ser una condición matricial, del tipo $A==B$, donde A y B son matrices del mismo tamaño. Para que se considere que la condición se cumple, es necesario que sean iguales dos a dos todos los elementos de las matrices A y B. Basta que haya dos elementos diferentes para que las matrices no sean iguales, y por tanto las sentencias del if no se ejecuten.

Análogamente, una condición en la forma $A\sim B$ exige que todos los elementos sean diferentes dos a dos. Bastaría que hubiera dos elementos iguales para que la condición no se cumpliera.

En resumen:

`if A==B` exige que todos los elementos sean iguales dos a dos

`if A~B` exige que todos los elementos sean diferentes dos a dos.

2.5.8.-Operaciones sobre funciones.-

Para encontrar mínimos de funciones unidimensionales y n-dimensionales usamos, respectivamente, las funciones:

`fmin('nombre_funcion',a,b)`

`fmins('nombre_funcion',a,b)`

Para buscar el cero de una función unidimensional usamos:

`fzero('nombre_funcion',a)`

donde a es el punto cerca del cual se busca el cero. A la función `fzero` debe darse el nombre de una función hecha por el usuario cuando se la llama.

También puede utilizarse para encontrar dónde una función es igual a cualquier

constante.

La función `inline` transforma en función una cadena de caracteres. `g = inline(expresion)` Por ejemplo, `fmin(inline('cos(x)'),3,4)`

`solve('ecuacion','x')`

resuelve la ecuación en la variable x .

MatLab proporciona tres funciones para calcular numéricamente el área bajo una función sobre un rango finito: `trapz`, `quad` y `quad8`.

`trapz(x,y)` aproxima la integral bajo una función al sumar el área de los trapecoides formados con los puntos

Las funciones `quad` y `quad8` realizan aproximaciones de un orden más elevado que un simple trapecoide, mediante la regla de Simpson. Funcionan igual que `fzero`.

2.5.9.-Lectura y grabación de datos.-

Supongamos un fichero llamado `datos.txt` que tiene, por ejemplo, dos columnas de valores que son números reales.

Para leer los datos del fichero:

```
[x1,x2] = textread('datos.txt','%f %f');
```

Probar

```
help textread
```

para otras opciones de formatos de entrada. Otra opción más:

```
load 'datos.txt' -ascii x1 = datos(:,1);
```

```
x2 = datos(:,2);
```


CAPITULO III
PROGRAMACIÓN GENERAL DEL SOFTWARE
COMPUTARIZADO M.E.F.

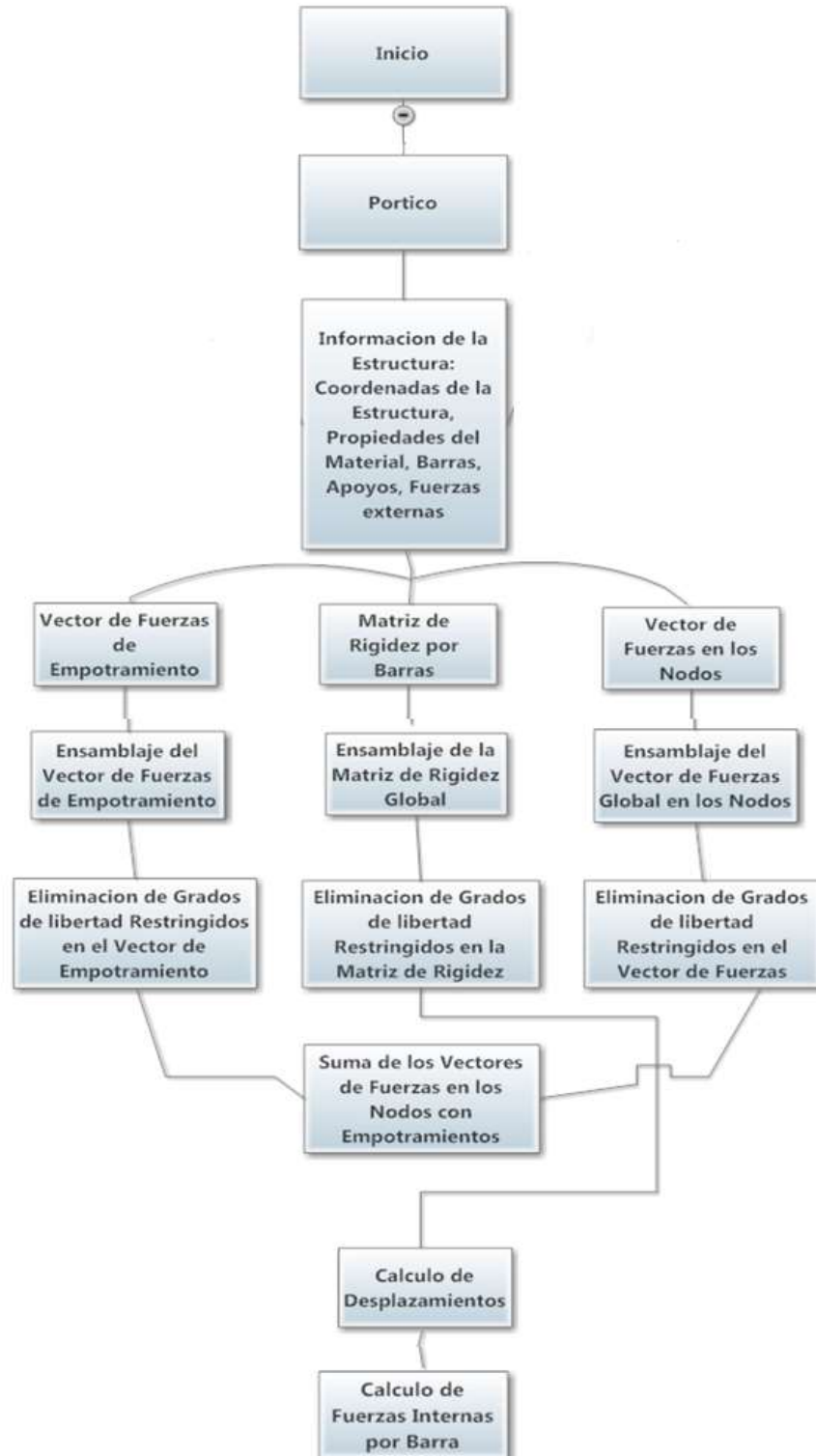
La aplicación con éxito en este caso del método de elementos finitos se reduce a un programa que fue elaborado con uso del lenguaje de programación MATLAB y con un sistema de funciones y scripts en Guide para la resolución de distintos tipos de problemas estructurales planos, estos conceptos se fueron desarrollando con teorías de elementos finitos de capítulos anteriores, el concepto que se elaboró para poder desarrollar el programa fue, en un sistema simple, gráfico y que tenga un uso netamente académico, ya que su fin es simplemente para uso docente-estudiante y para explicación en si del método paso a paso, ya que el lenguaje fue desarrollado de manera que pueda ser entendida para los estudiantes o docentes que tengan interés en el tema y puedan seguir investigando en el método. Se deja el código de programación abierto para posteriores estudiantes, docentes que tengan interés de investigación en este método y así poder ampliar su uso.

Toda la programación se realizó en el entorno grafico de MATLAB llamado Guide.

3.1.-Características del software M.E.F.-

En la ilustracion 22 se muestra el diagrama de flujo del programa, en el cual se analizará para los, sistemas o bucles más importantes de ella la cual nos dará una idea total del sistema. Para centrar conceptos definiremos seguidamente las etapas fundamentales asociadas al análisis de una estructura por un programa de elementos

finitos, así como la relación de cada etapa con las subrutinas puestas en el diagrama de flujo principal:



Ilustracion 22. Diagrama de flujo principal del Programa M.E.F.

3.2.-Inicio del programa computarizado M.E.F.-

La primera ventana que se visualiza al iniciar el programa M.E.F. es la rutina de Inicio (**Ilustracion 23**), en la cual se presenta el programa y se pone a elección el tipo de estructura a resolver, en nuestro caso portico, el código de programación de la Ventana de Inicio se brinda a continuación.

Ilustracion 23. Ventana de inicio

Código de programación de la ventana inicio:

```
function varargout = Inicio(varargin)
gui_Singleton = 1;
gui_State = struct('gui_Name',    mfilename, ...
                  'gui_Singleton', gui_Singleton, ...
                  'gui_OpeningFcn', @Inicio_OpeningFcn, ...
                  'gui_OutputFcn', @Inicio_OutputFcn, ...
                  'gui_LayoutFcn', [], ...
                  'gui_Callback', []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end
if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
```

```

else
    gui_mainfcn(gui_State, varargin{:});

end

function Inicio_OpeningFcn(hObject, eventdata, handles, varargin)
imshow(imread('UAJMS.png'));

set(handles.axes1,'xtick',[])
set(handles.axes1,'ytick',[])
handles.output = hObject;
guidata(hObject, handles);

function varargout = Inicio_OutputFcn(hObject, eventdata, handles)
varargout{ 1 } = handles.output;

function viga_Callback(hObject, eventdata, handles)
VIGA; %

function portico_Callback(hObject, eventdata, handles)
PORTICO; %

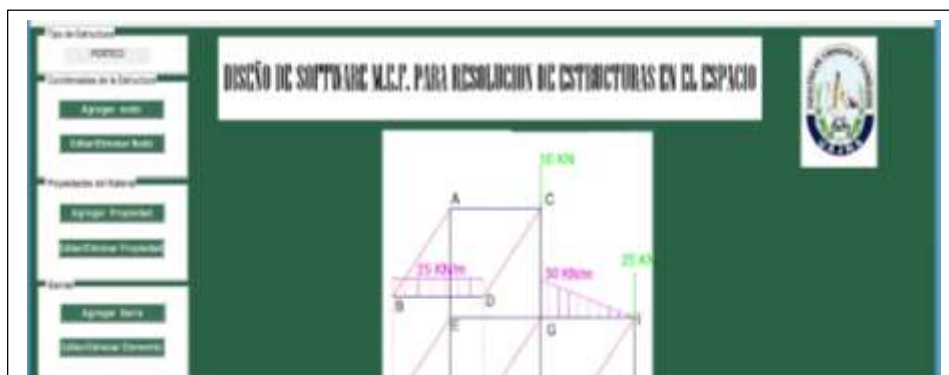
function armadura_Callback(hObject, eventdata, handles)
ARM;

% --- Executes on button press in salir.

```

3.3.-Ingreso de datos.-

Una vez escogido el tipo de estructura se abrirán ventanas según la elección (Ilustracion 24), el ingreso de datos para porticos en el espacio como ser : Coordenadas de la Estructura, Barras, Apoyos y Mostrar Resultados para poder simplificar el manejo del programa así como también poder reducir el código de programación para mayor efectividad y velocidad.



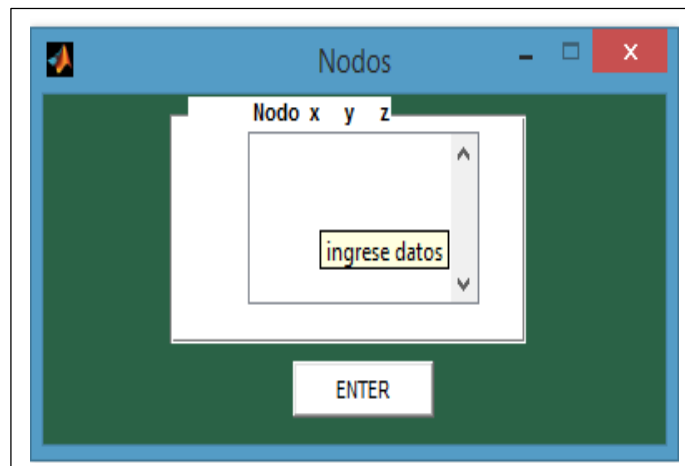
3.3.1.- Coordenadas de la estructura.-

La ventana de ingreso de coordenadas se divide en 2 partes, la primera es la de

Agregar Nodos y la segunda es la de Editar/Eliminar Nodos

Agregar nodos

Esta es la primera ventana que se visualiza para a introducción de datos, en dicha ventana se debe ingresar las coordenadas de cada nodo de la estructura, comenzando por la numeración del nodo, seguido de su coordenada en X ,Y, Z.



Ilustracion25. Ventana de Ingreso de Coordenadas.

La rutina encargada del ingreso de nodos fue llamada Nodos, se muestra a continuación su programación.

Codigo de Programacion de la ventana Ingreso de Nodos:

```

function varargout =
Nodos(varargin)

gui_Singleton
= 1;
gui_State = struct('gui_Name', mfilename, ...
                  'gui_Singleton', gui_Singleton, ...
                  'gui_OpeningFcn', @Nodos_OpeningFcn, ...
                  'gui_OutputFcn', @Nodos_OutputFcn, ...
                  'gui_LayoutFcn', [], ...
                  'gui_Callback', []);

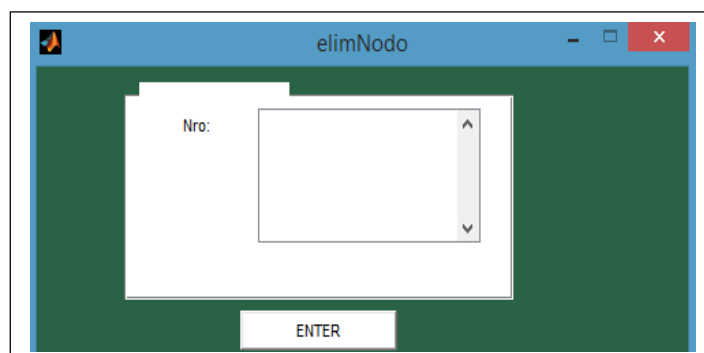
if nargin &&
ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

    if nargout
        [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
    else
        gui_mainfcn(gui_State, varargin{:});
    end
end

```

Editar/Eliminar nodos

Esta rutina como su nombre lo indica se encarga de la edición y eliminación de nodos, al igual que la ventana coordenadas se debe ingresar primeramente la numeración del nodo a editar/eliminar, seguido de su coordenada en X ,Y, Z.



La rutina encargada de editar/eliminar nodos fue denominada ElimNodo, se muestra a continuación su programación:

Código de programación de la ventana ingreso de Elim Nodo:

```
function varargout = elimNodo(varargin)

gui_Singleton
= 1;

gui_State = struct('gui_Name',    mfilename, ...
                  'gui_Singleton', gui_Singleton, ...
                  'gui_OpeningFcn', @elimNodo_OpeningFcn, ...
                  'gui_OutputFcn', @elimNodo_OutputFcn, ...

                  'gui_LayoutFcn', [], ...
                  'gui_Callback', []);

if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end

function elimNodo_OpeningFcn(hObject, eventdata, handles, varargin)
global M
set(handles.numero_nodo, 'string', num2str(M))
handles.output = hObject;
guidata(hObject, handles);

function varargout = elimNodo_OutputFcn(hObject, eventdata, handles)
```



```

varargout{1} = handles.output;

function numero_nodo_Callback(hObject,eventdata, handles)
function enter_Callback(hObject,eventdata, handles)
global M
a=dir('coordenada*.mat');
for N=1:size(a,1)
    delete(['coordenada',num2str(N),'.mat']);
end
M=str2num(get(handles.numero_nodo,'string'));
for N=1:size(M) P=M(N,:);
    Nombre=['coordenada',num2str(N),'.mat']
    ; save(Nombre,'P')
end
close('elimNo
do');

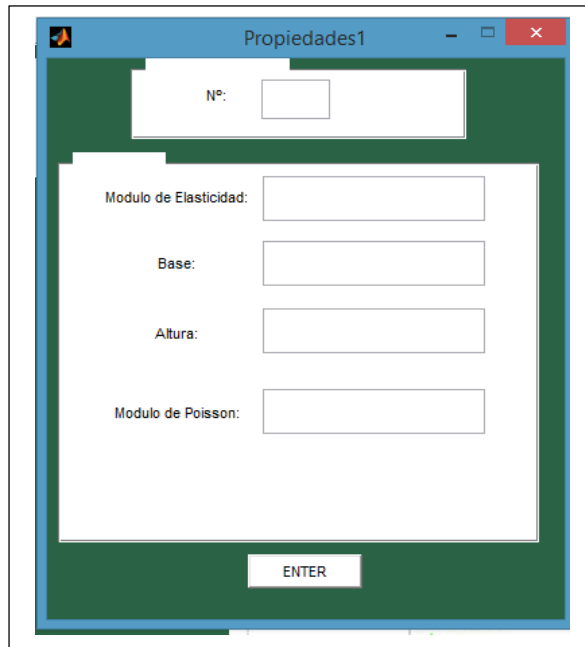
```

3.3.2.-Propiedades del material.-

Esta rutina se encarga de recopilar la información de las propiedades del material como ser, Inercia y Modulo de Elasticidad en el caso, en el caso de Pórticos; Área y Modulo de Elasticidad.

3.3.2.1.-Agregar propiedad.-

La rutina de Agregar Propiedad se denominó Numero de Propiedad.



Ilustracion 27. Agregar propiedad en pórticos.

Las rutinas encargadas de agregar propiedades del material contienen el siguiente código de programación.

3.3.3.-Barras.-

El panel “Barras” se encarga de recopilar información de las conexiones de las barras, la columna uno enumera la barra, en la columna dos se coloca el número de propiedad y en la columna 3 y 4 el nodo inicial y final.

3.3.3.1. -Agregar barra.-

La rutina de barras el nombre con el que se la denomino es Elementos, y tiene el

siguiente código de programación y la ventana se presenta en la ilustracion 28



Ilustracion 28. Ventana de ingreso de barras

Código de programación de la ventana agregar barras:

```
function varargout = Elementos(varargin)

gui_Singleton = 1;
gui_State = struct('gui_Name',    mfilename, ...
                  'gui_Singleton', gui_Singleton, ...
                  'gui_OpeningFcn', @Elementos_OpeningFcn, ...

varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end

function Elementos_OpeningFcn(hObject, eventdata, handles, varargin)
    a=dir('elemento*.mat');
    for N=1:size(a,1)
```

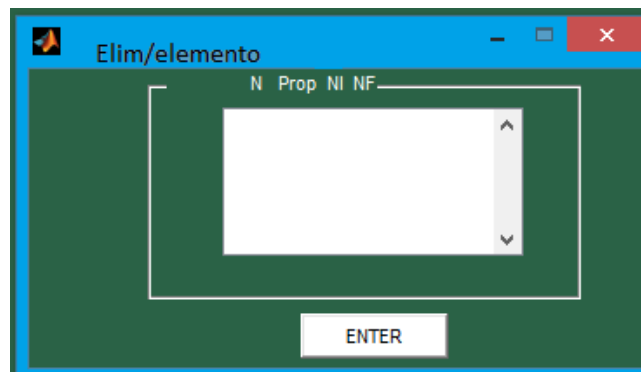
```

delete(['elemento',num2str(N),
handles.output;
function Numero_barras_Callback(hObject,eventdata,handles)
function enter_elementos_Callback(hObject,eventdata,handles)
global E
E=str2num(get(handles.Numero_barras,'string'));
for N=1:size(E) P=E(N,:);
    Nombre=['elemento',num2str(N),'.mat']
    ; save(Nombre,'P')
end
close('Elementos')

```

3.3.3.2.-Editar/Eliminar Barra.-

La rutina Editar/Eliminar Barra como su nombre lo dice se encarga de la edición o en su defecto la eliminación de barras, simplemente se debe colocar el número de barra que se desee eliminar en la figura 3.12 se presenta la ventana de Edición/Eliminación de Barras.



Ilustracion 29. Ventana de Editar/Eliminar barra

Esta rutina de denominó con el nombre de `ElimElemento.m` y su código de programación es el siguiente:

```
function varargout = elimElemento(varargin)

gui_Singleton
= 1;
gui_State = struct('gui_Name',    mfilename, ...
                  'gui_Singleton', gui_Singleton, ...
                  'gui_OpeningFcn', @elimElemento_OpeningFcn, ...
                  'gui_OutputFcn', @elimElemento_OutputFcn, ...
                  'gui_LayoutFcn', [], ...
                  'gui_Callback', []);

if nargin == 0
    ischar(varargin{1})
        gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end

= elimElemento_OutputFcn(hObject, eventdata, handles)

varargout{1} = handles.output

function eliminar_elemento_Callback(hObject, eventdata, handles)
function enter_Callback(hObject, eventdata, handles)

global E
a=dir('elemento*.mat'); for N=1:size(a,1)
    delete(['elemento',num2str(N),'.mat']);
```

```

end
E=str2num(get(handles.eliminar_elemento,'string'));
for N=1:size(E)
    P=E(N,:);
    Nombre=['elemento',num2str(N),'.mat'];
    save(Nombre,'P')
end
close('elimElemento');

'gui_Callback', []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if narginout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});

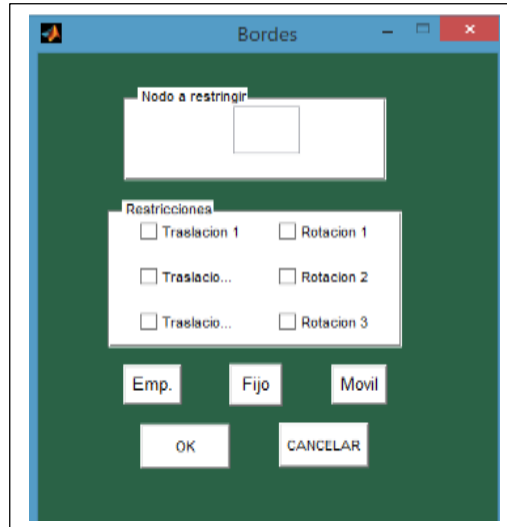
```

3.3.4.-Definir apoyos.-

La rutina de definir apoyos se encarga de recopilar la información correspondiente a los apoyos de la estructura, definiendo automáticamente los grados de libertad.

3.3.4.1.-Agregar apoyo.-

Esta ventana se encarga de guardar los tipos de apoyos, cabe recalcar que para el caso de cada uno de estos (empotrado, fijo o movil) se presenta tambien sus respectivas restricciones.



Ilustracion 30. Ventana de apoyos

Código de programación de la ventana definir apoyos:

```
function varargout = BordesA(varargin)
gui_Singleton = 1;
gui_State = struct('gui_Name',    mfilename, ...
                  'gui_Singleton', gui_Singleton, ...
                  'gui_OpeningFcn', @BordesA_OpeningFcn, ...
                  'gui_OutputFcn', @BordesA_OutputFcn, ...
                  'gui_LayoutFcn', [], ...
                  'gui_Callback', []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end
if nargin
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
```

```

function BordesA_OpeningFcn(hObject, eventdata, handles, varargin)
axes(handles.axes1)

imshow(imread('Fijo.jpg'));
set(handles.axes1,'xtick',[])
set(handles.axes1,'ytick',[])
handles.output = hObject;
axes(

function varargout = BordesA_OutputFcn(hObject, eventdata, handles)
varargout{1} = handles.output;

function Apoyo_fijo_Callback(hObject, eventdata, handles)
Nodo1=str2double(get(handles.N_nodo,'String'));
af=[0 0];

P=[Nodo1 af];
Nombre=['bordes',num2str(Nodo1),'.mat']
; save(Nombre,'P')
close('BordesA')

function apoyo_movil_Callback(hObject, eventdata, handles)
Nodo1=str2double(get(handles.N_nodo,'String'));
am=[ 1 0 ];

P

ispcc &&

end

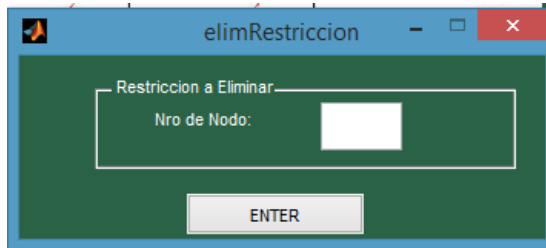
function axes1_CreateFcn(hObject, eventdata, handles)

function axes2_CreateFcn(hObject, eventdata, handles)

```

3.3.4.2.- Editar/Eliminar apoyos.-

La rutina de definir apoyos se encarga de editar y eliminar nodos que hayan sido introducidos por equivocación.



Ilustracion 31 Ventana de apoyos a eliminar

Codigo de programación de la rutina elim restriccion:

```
function varargout = elimRestriccion(varargin)
gui_Singleton = 1;
gui_State = struct('gui_Name',    mfilename, ...
                  'gui_Singleton', gui_Singleton, ...
                  'gui_OpeningFcn', @elimRestriccion_OpeningFcn, ...
                  'gui_OutputFcn', @elimRestriccion_OutputFcn, ...
                  'gui_LayoutFcn', [], ...
                  'gui_Callback', []);

if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

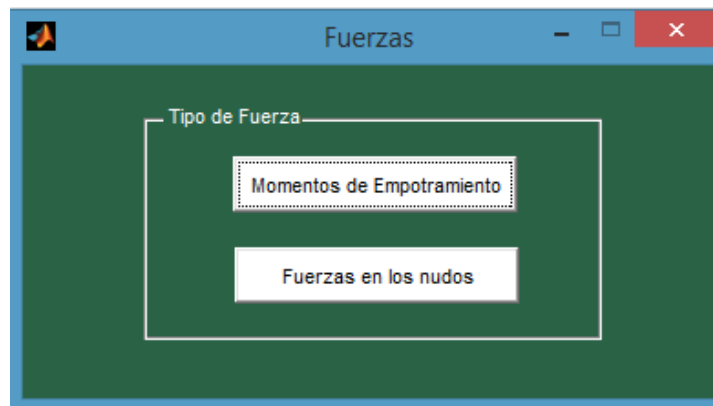
if nargin
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end

function elimRestriccion_OpeningFcn(hObject,eventdata, handles,
varargin)
handles.output = hObject;
guidata(hObject, handles);
```

```
function varargout = elimRestriccion_OutputFcn(hObject,eventdata,
handles)
varargout{1} = handles.output;
function eliminar_Restriccion_Callback(hObject,eventdata, handles)
er=str2double(get(handles.eliminar_Restriccion,'String'));
delete(['bordes',num2str(er),'.mat']);
```

3.3.5.- Fuerzas externas.-

En la ventana de fuerzas externas (Ilustracion 32) nos muestra la opción de insertar momentos de empotramiento y fuerzas ejercidas directamente en los nodos.



Ilustracion 32. Ventana de fuerzas.

3.3.5.1. Momentos de empotramiento.-

La rutina Momentos de empotramiento en Porticos muestra distintos estados de carga, que pudieran estar presentes en las barras. Simplemente se debe elegir el estado de carga que se tenga sobre la barra y se abrirá una ventana donde se calcularan los momentos de manera sencilla e intuitiva.

La rutina de esta ventana se denomina Casos Empotramiento.

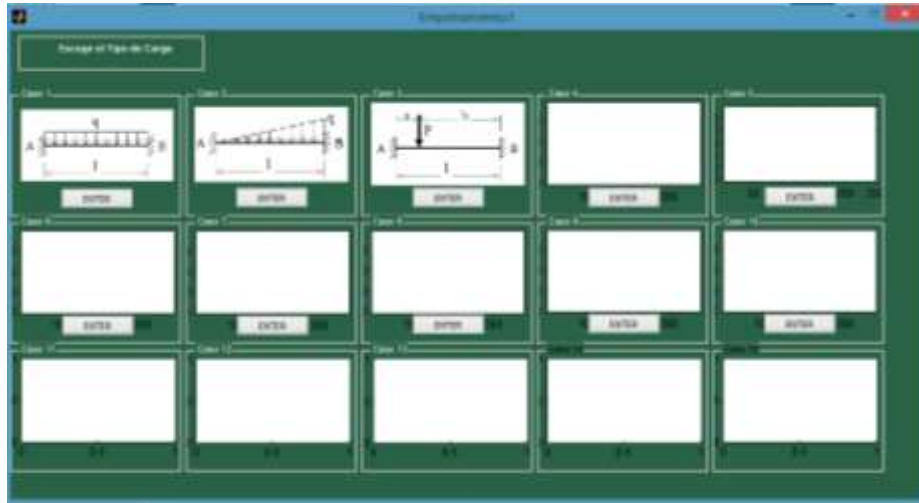


Ilustración 33 Ventana de casos de momentos de empotramiento.

Código de programación de la ventana Casos Empotramiento:

```
function varargout = CasoEmpotramiento(varargin)
gui_Singleton = 1;
gui_State = struct('gui_Name',    mfilename, ...
                  'gui_Singleton', gui_Singleton, ...
                  'gui_OpeningFcn', @CasoEmpotramiento_OpeningFcn, ...
                  'gui_OutputFcn', @CasoEmpotramiento_OutputFcn, ...
                  'gui_LayoutFcn', [], ...
                  'gui_Callback', []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end
if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
```

```
function CasoEmpotramiento_OpeningFcn(hObject,eventdata, handles,  
varargin)
```

```
axes(handles.axes1); imshow(imread('M1.png'));  
  
set(handles.axes1,'xtick',[]);set(handles.axes1,'ytick',[])  
axes(handles.axes2); imshow(imread('M5.png'));  
set(handles.axes2,'xtick',[]);set(handles.axes2,'ytick',[])  
axes(handles.axes3); imshow(imread('M9.png'));  
set(handles.axes3,'xtick',[]);set(handles.axes3,'ytick',[])  
handles.output = hObject;  
  
guidata(hObject, handles);
```

```
function varargout = CasoEmpotramiento_OutputFcn(hObject,eventdata,  
handles)
```

```
varargout{ 1 } = handles.output;
```

```
function
```

```
enter1_Callback(hObject, eventdata, handles)
```

```
function enter2_Callback(hObject, eventdata, handles)
```

```
enter3_Callback(hObject, eventdata, handles)
```

```
function enter4_Callback(hObject, eventdata, handles)
```

```
function enter5_Callback(hObject, eventdata, handles)
```

```
function enter6_Callback(hObject, eventdata, handles)
```

```
function enter7_Callback(hObject, eventdata, handles)
```

```
function enter8_Callback(hObject, eventdata, handles)
```

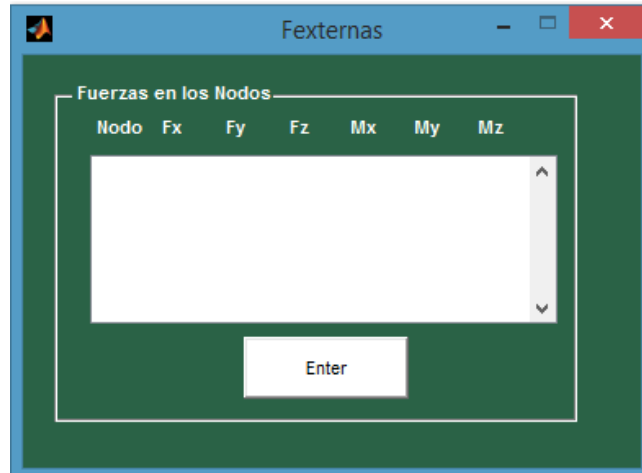
```
function enter9_Callback(hObject, eventdata, handles)
```

```
function enter10_Callback(hObject, eventdata, handles)
```

3.3.5.2-. Fuerzas en los nodos.-

Esta se encarga de recopilar datos sobre fuerzas que estén actuando directamente en los nodos, dicha rutina lleva el nombre de Fexternas y en la misma se debe ingresar

en la columna uno el número de nodo donde actuara la carga, en la columna 2, 3 y 4 se debe ingresar fuerzas y momentos.

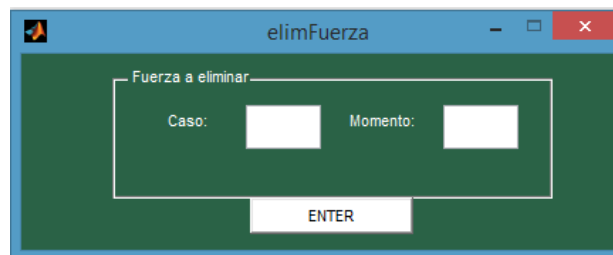


Ilustracio 34 Ventana de Fuerzas en los Nodos.

3.3.6. Editar/Eliminar Fuerza.-

En esta ventana se debe introducir el caso de estado de carga en la primera ventana y en la segunda ventana colocar la barra a la que corresponde el caso de empotramiento.

La rutina se llama elimFuerza. (Ilustracion 40), tiene el siguiente código de programación:



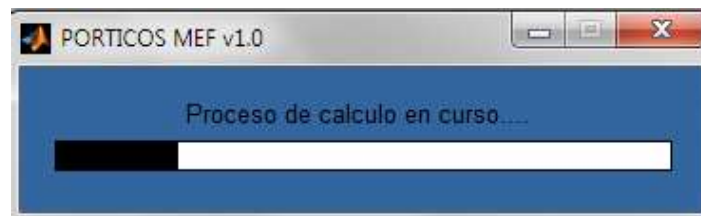
Ilustracion 35 Ventana de Editar/Eliminar fuerza

```
function varargout = elimFuerza(varargin)
gui_Singleton = 1;
gui_State = struct('gui_Name',    mfilename, ...
                  'gui_Singleton', gui_Singleton, ...
                  'gui_OpeningFcn', @elimFuerza_OpeningFcn, ...
                  'gui_OutputFcn', @elimFuerza_OutputFcn, ...
                  'gui_LayoutFcn', [] , ...
                  'gui_Callback', []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end
if nargin
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
function elimFuerza_OpeningFcn(hObject, eventdata, handles, varargin)
handles.output = hObject;
guidata(hObject, handles);
function varargout = elimFuerza_OutputFcn(hObject, eventdata, handles)
varargout{1} = handles.output;
function momento_Callback(hObject, eventdata, handles)
function caso_Callback(hObject, eventdata, handles)
function momento_CreateFcn(hObject, eventdata, handles)
function enter_eliminar_fuerza_Callback(hObject, eventdata, handles)
N=str2double(get(handles.momento,'String'));
c=str2double(get(handles.caso,'String'));
delete(['momentosMO',num2str(c),'_',num2str(N),'.mat']);
close('elimFuerza')
guidata(hObject, handles);
```

3.4. Calcular estructura.-

Esta rutina es una de las más importantes del programa puesto a que es en este apartado donde se realiza todo el cálculo de la estructura mediante el método de elementos finitos y mientras se realiza este cálculo, se va creando un archivo Excel donde se muestra el procedimiento de cálculo paso a paso, simplemente se debe presionar el botón “Calcular Estructura” y comienza el cálculo donde el progreso se puede visualizar en una

Barra (Ilustracion 36) que tiene el siguiente código de programación y va progresando a medida que se va realizando el cálculo.



Ilustracion 37 Barra de progreso de Calculo.

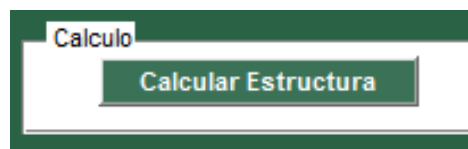
Codigo de programación de la barra de progreso:

```
wb=waitbar(0,'Proceso de calculo en curso....','color',[
    ]);
set(wb,'Name','PORTICOS MEF v1.0');

set(wbObject,'EdgeColor',[0 0 0],'FaceColor',[0 0 0]);
waitbar(10/100);
pause(1);
```

```
waitbar(20/100);
pause(1);
waitbar(30/100);
pause(1);
waitbar(40/100);
pause(1);
waitbar(50/100);
pause(1);
waitbar(60/100);
pause(1);
waitbar(70/100);
pause(1);
waitbar(80/100);
pause(1);
waitbar(90/100);
pause(1);
waitbar(100/100);
delete(wb)
```

La rutina recibe el nombre de funcion `calcular_estructura_Callback` (Ilustracion 38), a continuación se muestra el código de programación de porticos.



Ilustracion 38. Botón para Calculo de la estructura.

Código de programación de la rutina function calcular estructura Callback para el cálculo de pórticos:

```
function calcular_estructura_Callback(hObject,eventdata, handles)
clear all
delete('calculos.xlsx');
wb=waitbar(0,'Proceso de calculo en curso....','color',[ 0.2 0.4 0.62
]);

set(wb,'Name','PORTICOS MEF v1.0');
wbObject=findobj(wb,'Type','Patch');
set(wbObject,'EdgeColor',[ 0 0 0],'FaceColor',[ 0 0 0 ]); ngl=0;

a=dir('coordenada*.mat');

for i=1:length(a)
nombres=load(a(i).name);
V(i,:)=nombres.P
```

3.5.-Mostrar resultados.-

Esta rutina se encarga de mostrar un archivo Excel donde se muestra el procedimiento de cálculo paso a paso para así poder lograr un entendimiento completo de la forma de resolución de estructuras mediante el método de elementos finitos.

La rutina recibe como nombre function mostrar resultados Callback y tiene el siguiente código de programación:

Codigo de programación function mostrar resultados Callback:

```
function mostrar_resultados_Callback(hObject,eventdata, handles)
winopen( 'Calculos.xlsx' )

function estructura_Callback(hObject, eventdata, handles)
contenido=get(hObject,'String');
a=get(hObject,'Value');
texto=contenido(a);
switch
cell2mat(texto)

    case
'Estructura'
Graficas

    case 'Cortante'
Graficas_Cortantes_Por
tico

    case 'Momentos'
Graficas_Mo
mentos end

function varargout = Graficas(varargin)

gui_Singleton = 1;
gui_State = struct('gui_Name',    mfilename, ...
                  'gui_Singleton', gui_Singleton, ...
                  'gui_OpeningFcn', @Graficas_OpeningFcn, ...
                  'gui_OutputFcn', @Graficas_OutputFcn, ...
                  'gui_LayoutFcn', [] , ...
```

```
        'gui_Callback', []);  
if nargin && ischar(varargin{1})  
    gui_State.gui_Callback = str2func(varargin{1});  
end
```

CAPITULO IV

ANÁLISIS DE RESULTADOS

En este capítulo se muestran los resultados obtenidos una vez terminado el software se procede a su validación con un software de uso comercial, en este caso se utilizara el Sap2000 v14.0.0 para apreciar en qué medida los datos analizados concuerdan con dicho software.

4.1. Portico en tres dimensiones.-

Se analizara el portico con las siguientes carateristicas y dimensiones.-

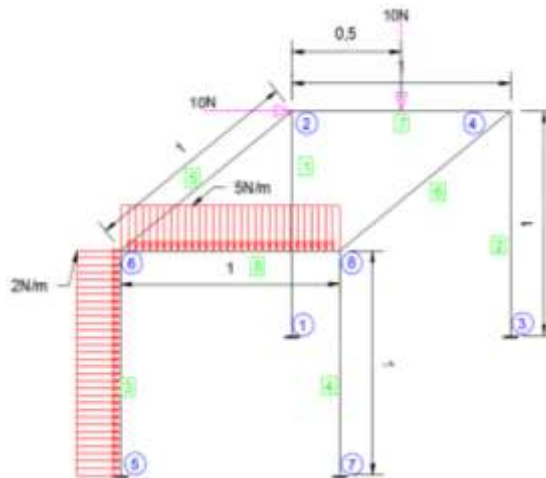
Modulo de elasticidad para todas las barras

$$E=2,00E10 \text{ Kg/m}^2$$

Seccion

$$a=0.2 \text{ m}$$

$$b=0.2 \text{ m}$$



Referencia:

Nº de Nudos	
Nº de Barras	
Cargas distribuida	
Cargas puntuales	

Eje			
Nudos	x	y	z
1	1	0	0
2	1	0	1
3	0	0	0
4	1	1	0
5	0	0	1
6	0	0	0
7	0	1	1
8	0	1	0

4.2.- Resultados por el software M.E.F.-

4.2.1.- Reacciones.-

Nudos	F1	F2	F3	M1	M2	M3
Reacciones						
	N	N	N	N-m	N-m	N-m
1	0.6760	-2.9800	2.3200	2.0800	0.3900	0.1570
3	-0.7000	-5.0000	7.6500	2.6000	-0.3800	0.1500
5	0.6800	-2.1300	0.7359	0.9189	0.3900	0.1579
7	-0.6700	-1.8800	4.2300	1.0000	-0.3800	0.1800

4.2.2.- Desplazamientos.-

Desplazamiento	U1	U2	U3	R1	R2	R3
Nudo	m	m	m	Radians	Radians	Radians
1	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
2	-0.0333	0.2035	-0.0029	-0.1978	-0.0186	-0.0901
3	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
4	0.0307	0.1960	-0.0096	-0.0371	0.0134	-0.0858
5	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
6	-0.0333	0.0777	-0.0009	-0.0783	-0.0186	-0.0901
7	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
8	0.0307	0.0766	-0.0054	-0.0212	0.0134	-0.0858

4.2.3.- Esfuerzos internos.-

FrameElem	F1	F2	F3	M1	M2	M3
Text	N	N	N	N-m	N-m	N-m
1 INICIAL	0.6780	-2.9800	2.3000	2.0800	0.3900	0.1588
1 FINAL	-0.6780	2.9800	-2.3000	0.9600	0.2900	-0.1588
2 INICIAL	-0.6780	-5.0500	7.6500	2.6200	-0.3800	0.1498
2 FINAL	0.67800	5.05000	-7.65000	2.43000	-0.31000	-0.14978
3 INICIAL	0.6780	-2.1000	0.7400	0.9200	0.3900	0.1601
3 FINAL	-0.6780	0.0975	-0.7400	0.1700	0.2900	-0.1601
4 INICIAL	-0.6780	-1.8800	4.2500	0.9900	-0.3800	0.1498
4 FINAL	0.6780	1.8800	-4.2500	0.8800	-0.3100	-0.1500
5 INICIAL	0.0000	1.0000	0.4778	-0.2100	-0.2400	0.5100
5 FINAL	0.0000	1.0000	0.4680	-0.2100	-0.2400	0.5100
6 INICIAL	0.0000	0.9600	-0.4997	-0.0276	0.2499	0.4800
6 FINAL	0.0000	-0.9600	0.5009	0.0276	0.2100	0.4800
7 INICIAL	0.6780	6.0100	1.8500	-0.7500	-0.0554	-0.3500
7 FINAL	-0.6780	-6.0100	8.1500	-2.4000	0.0554	-0.3300
8 INICIAL	0.6780	0.9200	1.2009	-0.3800	-0.0550	-0.3500
8 FINAL	-0.68	-0.92	3.79	-0.91	0.06	-0.33

4.3.- Resultados por el SAP v2000 16.-

4.3.1.- Reacciones.-

TABLE: Joint Reactions								
Joint	OutputCase	CaseType	F1	F2	F3	M1	M2	M3
Text	Text	Text	N	N	N	N-m	N-m	N-m
1	DEAD	LinStatic	0.68	-2.98	2.32	2.02	0.39	0.16
3	DEAD	LinStatic	-0.68	-5.05	7.65	2.62	-0.38	0.15
5	DEAD	LinStatic	0.68	-2.1	0.74	0.92	0.39	0.16
7	DEAD	LinStatic	-0.68	-1.88	4.29	0.99	-0.38	0.15

4.3.2.- Desplazamientos.-

TABLE: Joint Displacements			U1	U2	U3	R1	R2	R3
Joint	OutputCase	CaseType	U1	U2	U3	R1	R2	R3
Text	Text	Text	m	m	m	Radians	Radians	Radians
1	DEAD	LinStatic	0	0	0	0	0	0
2	DEAD	LinStatic	-0.033273	0.203533	-0.002901	-0.197761	-0.018582	-0.09005
3	DEAD	LinStatic	0	0	0	0	0	0
4	DEAD	LinStatic	0.030671	0.196026	-0.009565	-0.037143	0.013378	-0.085803
5	DEAD	LinStatic	0	0	0	0	0	0
6	DEAD	LinStatic	-0.033273	0.077731	-0.000922	-0.078306	-0.018582	-0.09005
7	DEAD	LinStatic	0	0	0	0	0	0
8	DEAD	LinStatic	0.030671	0.076582	-0.005362	-0.021234	0.013378	-0.085803

4.3.3.- Esfuerzos internos.-

TABLE: Element Joint Forces - Frames										
Frame	Joint	OutputCase	CaseType	F1	F2	F3	M1	M2	M3	FrameElem
Text	Text	Text	Text	N	N	N	N-m	N-m	N-m	Text
1	1	DEAD	LinStatic	0.68	-2.98	2.32	2.02	0.39	0.16	1
1	2	DEAD	LinStatic	-0.68	2.98	-2.32	0.96	0.29	-0.16	1
2	3	DEAD	LinStatic	-0.68	-5.05	7.65	2.62	-0.38	0.15	2
2	4	DEAD	LinStatic	0.68	5.05	-7.65	2.43	-0.31	-0.15	2
3	5	DEAD	LinStatic	0.68	-2.1	0.74	0.92	0.39	0.16	3
3	6	DEAD	LinStatic	-0.68	0.09753	-0.74	0.17	0.29	-0.16	3
4	7	DEAD	LinStatic	-0.68	-1.88	4.29	0.99	-0.38	0.15	4
4	8	DEAD	LinStatic	0.68	1.88	-4.29	0.88	-0.31	-0.15	4
5	2	DEAD	LinStatic	0	1.02	0.47	-0.21	-0.24	0.51	5
5	6	DEAD	LinStatic	0	-1.02	-0.47	0.21	-0.24	0.51	5
6	4	DEAD	LinStatic	0	0.96	-0.5	-0.02758	0.25	0.48	6
6	8	DEAD	LinStatic	0	-0.96	0.5	0.02758	0.25	0.48	6
7	2	DEAD	LinStatic	0.68	6.01	1.85	-0.75	-0.0554	-0.35	7
7	4	DEAD	LinStatic	-0.68	-6.01	8.15	-2.4	0.0554	-0.33	7
8	6	DEAD	LinStatic	0.68	0.92	1.21	-0.38	-0.0554	-0.35	8
8	8	DEAD	LinStatic	-0.68	-0.92	3.79	-0.91	0.0554	-0.33	8

4.4.- Analisis Comparativo.-

Nudos	F1	F2	F3	M1	M2	M3
Reacciones	N	N	N	N-m	N-m	N-m
1(SAP)	0.6800	-2.9800	2.3200	2.0200	0.3900	0.1600
1(MEF)	0.6760	-2.9800	2.3200	2.0800	0.3900	0.1570
Margen de Error	0.59	0.00	0.00	2.97	0.00	0.98
3(SAP)	-0.6800	-5.0500	7.6500	2.6200	-0.3800	0.1500
3(MEF)	-0.7000	-5.0000	7.6500	2.6000	-0.3800	0.1500
Margen de Error	2.94	0.99	0.00	0.76	0.00	0.00
5(SAP)	0.6800	-2.1000	0.7400	0.9200	0.3900	0.1600
5(MFE)	0.6800	-2.1300	0.7359	0.9189	0.3900	0.1579
Margen de Error	0.00	1.43	0.55	0.12	0.00	0.99
7(SAP)	-0.6800	-1.8800	4.2900	0.9900	-0.3800	0.1800
7(MEF)	-0.6700	-1.8800	4.2300	1.0000	-0.3800	0.1800
Margen de Error	1.47	0.00	1.40	1.01	0.00	0.00

Desplazamiento	U1	U2	U3	R1	R2	R3
Nudo	m	m	m	Radians	Radians	Radians
1(SAP)	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
1(MEF)	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
Margen de Error	0.00	0.00	0.00	0.00	0.00	0.00
2(SAP)	-0.033273	0.203533	-0.002901	-0.197761	-0.018582	-0.09005
2(MEF)	-0.0333	0.2035	-0.0029	-0.1978	-0.0186	-0.0901
Margen de Error	0.08	0.02	0.03	0.20	0.10	0.06
3(SAP)	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
3(MEF)	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
Margen de Error	0.00	0.00	0.00	0.00	0.00	0.00
4(SAP)	0.0307	0.1960	-0.0096	-0.0371	0.0134	-0.0858
4(MEF)	0.0307	0.1960	-0.0096	-0.0371	0.0134	-0.0858
Margen de Error	0.09	0.01	0.37	0.12	0.16	0.00
5(SAP)	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
5(MEF)	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
Margen de Error	0.00	0.00	0.00	0.00	0.00	0.00
6(SAP)	-0.0333	0.0777	-0.0009	-0.0783	-0.0186	-0.0901
6(MEF)	-0.0333	0.0777	-0.0009	-0.0783	-0.0186	-0.0901
Margen de Error	0.08	0.04	2.31	0.01	0.10	0.06
7(SAP)	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
7(MEF)	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
Margen de Error	0.00	0.00	0.00	0.00	0.00	0.00
8(SAP)	0.0307	0.0766	-0.0054	-0.0212	0.0134	-0.0858
8(MEF)	0.0307	0.0766	-0.0054	-0.0212	0.0134	-0.0858
Margen de Error	0.09	0.02	0.71	0.16	0.16	0.00

FrameElem	F1	F2	F3	M1	M2	M3
Text	N	N	N	N-m	N-m	N-m
1 INICIAL (SAP)	0.6800	-2.9800	2.3200	2.0200	0.3900	0.1600
1 INICIAL (MEF)	0.6780	-2.9800	2.3000	2.0800	0.3900	0.1588
Margen de Error	0.29	0	0.86	2.97	0	0.76
1 FINAL(SAP)	-0.6800	2.9800	-2.3200	0.9600	0.2900	-0.1600
1 FINAL (MEF)	-0.6780	2.9800	-2.3000	0.9600	0.2900	-0.1588
Margen de Error	0.29	0	0.86	0	0	0.76
2 INICIAL (SAP)	-0.6800	-5.0500	7.6500	2.6200	-0.3800	0.1500
2 INICIAL (MEF)	-0.6780	-5.0500	7.6500	2.6200	-0.3800	0.1498
Margen de Error	0.29	0	0	0	0	0.15
2 FINAL (SAP)	0.68000	5.05000	-7.65000	2.43000	-0.31000	-0.15000
2 FINAL (MEF)	0.67800	5.05000	-7.65000	2.43000	-0.31000	-0.14978
Margen de Error	0.29	0	0	0	0	0.15
3 INICIAL (SAP)	0.6800	-2.1000	0.7400	0.9200	0.3900	0.1600
3 INICIAL (MEF)	0.6780	-2.1000	0.7400	0.9200	0.3900	0.1601
Margen de Error	0.29	0	0	0	0	0.06
3 FINAL (SAP)	-0.6800	0.0975	-0.7400	0.1700	0.2900	-0.1600
3 FINAL (MEF)	-0.6780	0.0975	-0.7400	0.1700	0.2900	-0.1601
Margen de Error	0.29	0.03	0	0	0	0.06
4 INICIAL (SAP)	-0.6800	-1.8800	4.2900	0.9900	-0.3800	0.1500
4 INICIAL (MEF)	-0.6780	-1.8800	4.2500	0.9900	-0.3800	0.1498
Margen de Error	0.29	0	0.93	0	0	0.15
4 FINAL (SAP)	0.6800	1.8800	-4.2900	0.8800	-0.3100	-0.1500
4 FINAL (MEF)	0.6780	1.8800	-4.2500	0.8800	-0.3100	-0.1500
Margen de Error	0.29	0	0.93	0	0	0.02
5 INICIAL (SAP)	0.0000	1.0200	0.4700	-0.2100	-0.2400	0.5100
5 INICIAL (MEF)	0.0000	1.0000	0.4778	-0.2100	-0.2400	0.5100
Margen de Error	0	1.96	1.66	0	0	0
5 FINAL (SAP)	0.0000	1.0200	0.4700	-0.2100	-0.2400	0.5100
5 FINAL (MEF)	0.0000	1.0000	0.4680	-0.2100	-0.2400	0.5100
Margen de Error	0	1.96	0.43	0	0	0
6 INICIAL(SAP)	0.0000	0.9600	-0.5000	-0.0276	0.2500	0.4800
6 INICIAL(MEF)	0.0000	0.9600	-0.4997	-0.0276	0.2499	0.4800
Margen de Error	0	0	0.06	0	0.05	0
6 FINAL (SAP)	0.0000	-0.9600	0.5000	0.0276	0.2500	0.4800
6 FINAL (MEF)	0.0000	-0.9600	0.5009	0.0276	0.2100	0.4800
Margen de Error	0	0	0.17	0	0.87	0
7 INICIAL (SAP)	0.6800	6.0100	1.8500	-0.7500	-0.0554	-0.3500
7 INICIAL (MEF)	0.6780	6.0100	1.8500	-0.7500	-0.0554	-0.3500
Margen de Error	0.29	0	0	0	0	0
7 FINAL (SAP)	-0.6800	-6.0100	8.1500	-2.4000	0.0554	-0.3300
7 FINAL (MEF)	-0.6780	-6.0100	8.1500	-2.4000	0.0554	-0.3300
Margen de Error	0.29	0	0	0	0	0
8 INICIAL (SAP)	0.6800	0.9200	1.2100	-0.3800	-0.0554	-0.3500
8 INICIAL (MEF)	0.6780	0.9200	1.2009	-0.3800	-0.0550	-0.3500
Margen de Error	0.29	0	0.75	0	0.72	0
8 FINAL (SAP)	-0.68	-0.92	3.79	-0.91	0.06	-0.33
8 FINAL (MEF)	-0.68	-0.92	3.79	-0.91	0.06	-0.33
Margen de Error	0	0	0	0	0	0

4.5.- Conclusiones.-

- ✚ Se pudo realizar el software con resultados de entre 0.00 y 3.00% comparado con el Sap2000 por lo tanto se puede concluir que los programas comerciales trabajan con el método de elementos finitos y la hipótesis ha sido resuelta como cierta.
- ✚ Las mismas estructuras siendo analizadas en otros programas como el programa Larch y el programa VigaG que son programados por elementos finitos concuerdan con el programa M.E.F. en un 100% dejando un 0% de porcentaje de error.
- ✚ El código que se brindó en el capítulo 3 es el código más óptimo que se pudo programar para llegar a los resultados tan concordantes como se pudo ver en el capítulo 4.
- ✚ Se presentaron todos los cálculos realizados en una hoja de Excel en donde se muestra paso a paso el procedimiento mediante elementos finitos que se utilizó para llegar a resultados tan exactos y confiables, y a través de funciones de comando (MAX, MIN, INDICE COICIDIR), obtenemos los valores máximos y mínimos.
- ✚ Se logró cumplir con el objetivo de brindar un software intuitivo con una interfaz gráfica amigable que permite poder aprender rápidamente la forma de ingreso de datos para así entender más rápidamente el funcionamiento del método de Elementos Finitos.

- ✚ Se deja el presente proyecto de grado, todo el código de programación utilizado para la realización del programa M.E.F. abiertamente para que futuros estudiantes que puedan mejorar o implementar nuevos tipos de estructuras como en su caso podría ser el de Placas planas, celosías espaciales, etc.
- ✚ Con el desarrollo de este proyecto de grado se aporta al estudio de software de análisis de elementos finitos que viene tomando forma en los últimos años mediante el desarrollo de otros problemas específicos de resistencia de materiales, para en un futuro proyecto reunir todos los trabajos y crear un software que involucre una solución numerosa de problemas y brindar así un buen uso educativo y académico, e impulsar mediante el estudio de casos unidimensionales, bidimensionales, tridimensionales a un estudio amplio y completo.

4.6.- Recomendaciones.-

- ✚ Se debe incentivar a la carrera en la creación de softwares cada vez más complejos y no esperar a que se desarrollen softwares de terceros que bien podrían ser desarrollados por estudiantes con la ayuda de docentes de nuestra superior casa de estudios U.A.J.M.S.
- ✚ El estudiante tanto así como el profesional deben entender a cabalidad cómo funcionan los programas que se están utilizando como por ejemplo es Sap2000 que es también un software basado en Elementos Finitos.

✚ Es un incentivo que las investigaciones con elementos finitos sigan adelante ya que es una gran herramienta para poder calcular problemas complejos en la ingeniería.

✚ Como actividades de incentivo se deberían realizar pequeños talleres, cursos y ponencias donde se muestre la importancia de este método en la actualidad, asimismo se debería profundizar más en el desarrollo de software mediante este método para resolver problemas académicos, esto con ayuda de las autoridades para que puedan financiar dichos cursos y ponencias